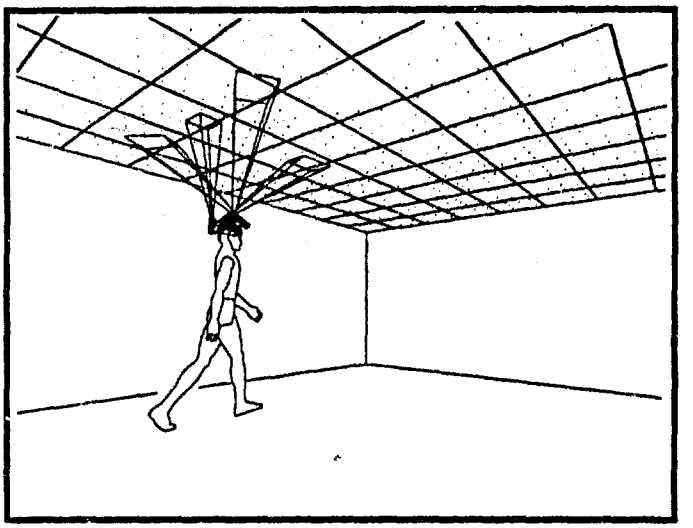
(2)

Advanced Technology for Portable Personal Visualization

Report of Research Progress
January – June 1992

This research is supported in part by
DARPA ISTO Contract No. DAEA 18-90-C-0044

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175
Telephones: 919-962-1911 (Fuchs), 919-962-1931 (Brooks)
FAX: 919-962-1799
E-mail: fuchs@cs.unc.edu, brooks@cs.unc.edu



Principal Investigators: Frederick P. Brooks, Jr., and Henry Fuchs

Faculty: Steve Pizer, Kenan Professor; Vern Chi, Director, Microelectronic Systems Lab; John Eyles, Research Assistant Professor; Gary Bishop, Research Associate Professor; Doug Holmgren, Visiting Assistant Professor, William V. Wright, Research Professor

Staff: Warren Robinett (Senior Researcher), Jannick Rolland (Optical Engineer), Steve Brumback (Electrical Engineer), David Harrison (Hardware Specialist, Video), John Hughes (Hardware Specialist, Force Feedback ARM), Linda Houseman (Administration), Kathy Tesh (Secretary), Fay Ward (Secretary)

Research Assistants: John Alspaugh (Architectural Walkthrough), Ron Azuma (Optical Tracker), Mike Bajura (Ultrasound), Andrew Bell (Walkthrough), Andy Brandt (Ultrasound), David Chen (Ultrasound), Jim Chung (Team Leader, Graphics Software), Drew Davidson (Sound User Interface), Erik Erikson (Molecular Graphics), Stefan Gottschalk (Tracker Simulation Software), Rich Holloway (Graphics Library), Phil Jacobsen (Optical Self-Tracker), Charles Kurak (Ultrasound), Mark Mine (Bathysphere), Uwe Nimscheck (Visiting RA, Architectural Walkthrough), Ryutarou Ohbuchi (Ultrasound), Russell Taylor (Nanomanipulator), Amitabh Varshney (Architectural Walkthrough), Yulan Wang (Architectural Walkthrough), Hans Weber (Architectural Walkthrough)

Advanced Technology for Portable Personal Visualization

1.0 Summary

Despite the recent avalanche of publicity on virtual reality, the state of the art in head-mounted displays is still at the "toy" stage. Although extravagant claims are made almost daily on television and in the popular press, there is very little that can be usefully accomplished today with virtual reality technologies.

We at UNC have been working since the early 1970s on aspects of these technologies and have been advancing the state of the art by a "driving problem" approach; we let the needs of selected applications stimulate the direction of the technological developments and then test new results by their impact on solving the original application. We have been working on three application areas: molecular modeling, 3D medical imaging, and modeling of architectural interiors.

We have developed complete systems that include the head-mounted display device, the display generation hardware, and a head and hand tracker. We purchased components when available and built components when there was a clear advantage to doing so:

a) We built, under separate major funding by DARPA and NSF, the display-generation hardware (our most recent machine is Pixel-Planes 5).
b) We built an optoelectronic helmet-tracking system (the ceiling tracker) that can determine position and orientation over a large area (currently under a 10- x 12-foot ceiling)—to our knowledge, the first demonstrated large-area helmet tracker.
c) We purchase commercially-available head-mounted display devices but continue to build head-mounted display devices with see-through capability.

As a result of collaboration with Professor John Hughes of Brown University and Professor Al Barr of California Institute of Technology, we have made great strides during the past six months in the self-calibration of our ceiling tracking technology. This may allow us to produce a large (20' x 20') ceiling tracker at a cost much cheaper than the current ceiling tracker.

A new optical see-through head-mounted display became operational during this reporting period. Through the development of see-through head-mounted display technology, we hope to give the wearer the ability to observe virtual-world objects superimposed onto his/her physical surroundings.

One highlight of this reporting period was our participation in the public television series, "The Machine That Changed the World." This five-part series, produced by WGBH-TV in Boston, dealt with the history of the computer. This series tried to chronicle the milestones in the development of the computer, to demonstrate how the computer is used for business and research today, and to touch on some possible uses for the future. A segment on virtual reality ran approximately 15 minutes, and almost all of it featured UNC's research in this area.

1.1 Goals of the Head-Mounted Display Project

- Demonstrate the usefulness of the head-mounted display (HMD) in real applications.
- Improve the hardware subsystems which currently limit performance of HMD (tracker, HMD optics and display, real-time graphics generation).
- Design and implement software base to support HMD functions.
- Integrate visual, auditory and haptic (force feedback) displays into a working system.
- Build new input devices and investigate methods of manual control suitable to a head-mounted display.
- Build and investigate the uses and limitations of see-through HMDs (optical and merged-video designs).

1.2 Goals of the See-through Head-Mounted Display Subproject

- Derive a method to accurately calibrate the see-through head-mounted display.
- Assess perception of depth and sizes using a bench prototype see-through head-mounted display.
- Build and integrate a 60-degree field-of-view system.
- Design, build, and integrate a new see-through head-mounted display using two miniature video cameras.

1.3 Goals of the Tracker Subproject

- Develop long-range trackers for head-mounted displays. Improve our existing tracking system, which can track an HMD inside a room-sized environment. For the long term, explore technologies that have unlimited range without requiring modification of the environment and that can track multiple users.
- Work to improve the other aspects of HMD tracking: latency, speed, accuracy, and resolution.

1.4 Goals of the Walkthrough Subproject

- Explore virtual environments through architectural simulation. Walking through a building is a natural and intuitive process in the real-world; the goal of the Walkthrough project is to use this intuitiveness to study virtual-world interaction.
- Serve as a driving problem for HMD and tracker research; Walkthrough is a real-world application that pushes the limits of HMD and tracker technologies.

2.0 Summary of Major Accomplishments

- At the recent 1992 Symposium on Interactive 3D Graphics in Cambridge, Massachusetts, there were 5 papers from UNC-CH on HMD-related applications (there were 8 papers in all from UNC on various graphics applications out of a total of 30 papers for the entire symposium). These HMD papers are listed in Section 5.1, and information on their presentation appears in Section 5.2.

2.1 Head-Mounted Display System

- During this reporting period, we have begun developing two new applications of head-mounted display:

  1) We have begun using a head-mounted display to merge ultrasound imagery with the real-world view.

  2) We have interfaced a scanning tunneling microscope to our HMD system to allow real-time viewing and exploration of nanometer-scale features on the surface of the sample beneath the microscope.

- VTK, a new virtual-worlds control panel/toolkit library, has been created to add more power and control to virtual-worlds applications. The toolkit provides a high-level interface for virtual sliders, knobs, and buttons in Vlib applications.

- We have integrated a new type of hand-held input device that is more natural and has more input capabilities than previous input devices.

- We have written a library to support using built-in serial lines and the phone switching system in Sitterson Hall, the building that houses the Department of Computer Science, for communication with serial devices, rather than running dedicated lines for each device.

- We are currently writing a library to control video signals for Pixel-Planes 5 applications so that switching between frame buffers can be done automatically via software.

- A new, faster commercial tracker has been integrated into trackerlib under a beta-test agreement. We have verified the manufacturer's claim of a 120 Hz update rate (which is 50% faster than our fastest comparable tracker); lag testing will be done soon.

- Many HMD applications have been upgraded and are becoming useful, real tools for users both inside and outside the HMD project. Both the 3D modeler (3DM) and Vixen have undergone several revisions in order to make them more useful for real work. Two Pixel-Planes 5 programs, Vixen and Xfront, were actually used for visualizing and debugging the design for the new 60-degree field-of-view see-through HMD. Vixen and 3DM have added a mechanism for saving still-frame images from interactive sessions to disk for later printout as slides.

- The number of users and applications of the HMD system continues to grow. Software engineering team projects investigated the use of HMDs for exploring animation, n-body simulations, and exploration of arbitrary surfaces.

- An application exploring the usefulness of variable-resolution text has been created. Models for the text are stored at several different levels of complexity in order to achieve interactive rates for large polygon counts.

- A technical report on the *viper* application (a near-real-time HMD application that optimizes interactive response at the expense of image quality) was published and made available to the public (See Appendix F).

- Jim Chung, who is working on the radiation treatment beam planning tool, presented the results of a preliminary experiment comparing different steering modes in a simplified beam-targeting task at the 1992 Symposium on Interactive Graphics. The experiment revealed minor differences between steering modes that used head-tracking information and modes that didn't. Inter-subject variability proved to be much greater than variation in the steering modes.

- We began development of the "bathysphere," a head-tracked stereo application using conventional high-resolution monitors in place of head-mounted displays. This involves a multiscreen environment around the user, who wears stereo glasses and has his/her head position and movement tracked. This system produces a simulated environment surrounding the user that gives the impression similar to the ocean surrounding a bathysphere. Initial results have motivated additional work toward the minimization of system latencies. For further discussion of this, see the sections on tracking.

- In January 1992 our experimental virtual-environment ultrasound scanning system was used by an ultrasound technician to scan a pregnant woman's fetus (See Appendix B).

2.2 See-through Head-Mounted Display with Custom Optics

- We have built and integrated a first prototype of the 30-degree field-of-view optical and mechanical head-mounted display system.

- The optical bench prototype for studying human factors has been completed, and a series of perceptual experiments designed.

- The optical bench prototype for studying human factors became operational in April 1992 and has been used for the last two months to study calibration problems as well as perception issues in an HMD via user studies.

- We have implemented on our computer graphics machine, Pixel-Planes 5, a mockup of the 60-degree field-of-view system on a digitized head of one of the team members (Fuchs) in order to study the geometry of the system, the distribution of the weight of the optics relative to the head, and the risks of misalignments that may occur as a result of frequent use by different researchers.

- Two decisions were made following the 3D mockup: 1) to modify the design to fold the miniature CRTs so that they come into alignment with the helmet. This will result in a more rigid, compact, and reliable system, and 2) to increase the effective eye relief as much as possible to allow for any user's glasses.

- We are collaborating with Tektronix to build miniature color CRTs that will be used as displays for the 60-degree field-of-view optics.

- We have completed the optical specification for a wide-angle see-through HMD using miniature video cameras and have begun designing the lenses for the video cameras.

2.3 Tracking

Background:

First demonstrated at the ACM SIGGRAPH '91 conference (28 July–2 August 1991), our custom-built optical tracking system features a scalable work area that currently measures 10' x 12'. The sensors consist of four head-mounted imaging devices that view infrared light-emitting diodes (LEDs) mounted in standard size (2' x 2') suspended ceiling panels. Photogrammetric techniques allow the head's location to be expressed as a function of the known LED positions and their projected images on the sensors. Discontinuities that occurred when changing working sets of LEDs were reduced by carefully managing all error sources, including LED placement tolerances, and by adopting an overdetermined mathematical model for the computation of head position: space resection by collinearity. A novel aspect of this system is that the range is not limited to the current configuration. By adding more panels to the ceiling grid, we can scale the system to any desired room size. To our knowledge, this is the first demonstrated scalable tracking system for HMDs.

After its introduction last summer, we installed this optical tracking system in our graphics laboratory, where it is now a platform for further tracker research and for running HMD applications. It supports the Walkthrough project, which retired their treadmill input device in favor of our optical tracker, and the Ultrasound group also intends to start using this system for their application.

Achievements during this reporting period:

- We have demonstrated the effectiveness of a simple calibration algorithm for refining the measurements of the locations of the ceiling's LEDs. The ability to measure LED locations in the ceiling means we will no longer have to align carefully the LED positions to a known grid. This will greatly reduce the expense needed to expand the ceiling to around 20' x 20'.

- Improvements in the low-level code have increased the update rate and reduced the latency for the optical tracker. Assuming the user keeps at least three sensors aimed at the ceiling, the update ranges from 30–150+ Hz, and the latency ranges from 10–50 ms. Typical values are 80–110 Hz and 15–30 ms. We have recently added code to limit the maximum update rate to 100 Hz to avoid stressing the LEDs in the ceiling too hard. Improvements in the heuristics used to choose which LEDs to light have reduced the variability in the average update rate, thus improving overall performance. They have also increased the number of discontinuities in the tracker output, so more work needs to be done on these heuristics.

- We implemented some simple predictive tracking routines in simulation and ran through these simulation routines actual head motion data collected from the ceiling tracker. Work is underway to implement these on our ceiling tracker system. Doing this properly requires accurate control of time, down to a few milliseconds, and requires us to remove ourselves completely from all non–real-time environments, such as UNIX. We hope that prediction will turn out to be an effective means of compensating for the delay inherent in virtual-environment systems.

- We verified the ceiling tracker's resolution specification of 0.2 degrees in orientation and 2 mm in position for short-range movements. We built a measurement rig, accurate to less than 1 millimeter and 1/60 of a degree. This rig contains clamps that hold the head unit rigidly. We can now move the head unit a specified amount and see how closely the tracker outputs match the distance as measured by the rig.

- To reduce the weight of the head unit, we replaced the existing lenses with a new set of smaller, lighter lenses. This substitution saved about 2 pounds of weight. Current head unit weight is about 8.5 lbs. Of that weight, 2 pounds are from the display, 2 pounds are from the four sensors and lenses, and the remainder comes from the supporting frame and cables.
- We also modified the calibration routine used to measure the distortion in the new lenses. The more accurate calibration visibly reduced the amount of discontinuities in the tracker output.
- We added routines to put the tracker into a "hibernating" mode if it is left unattended for a long time. This permits us to leave the tracker running all of the time, while avoiding putting undue stress on the LEDs in the ceiling.
- Recording capabilities, which include timing and position, have been extended to record accurately the latency in the various modules of the tracker and the number of LEDs seen in each iteration.
- We presented a paper describing the optical tracker (see Sections 5.1 and 5.2).
- Dr. John F. Hughes of Brown University visited us again to work with us on calibration techniques for the ceiling and the Polhemus we use for hand tracking.
- Dr. Michael Moshell, from the University of Central Florida, visited us to use the ceiling tracker in an experiment in unlimited distance walking with trackers that have limited range. The basic idea is to rotate the virtual world slowly as the user moves so that, although he feels as if he is walking in a straight line, in reality he walks in circles, always staying inside tracker range. The results from this preliminary experiment were surprisingly encouraging.
- Hand tracking with a Polhemus source attached to the head unit was demonstrated in several applications, involving operations such as grabbing, scaling, and moving objects. To reduce the distortion in the magnetic field, we are experimenting with a "unicorn mount" that moves the Polhemus source further away from the head unit.
- We accurately measured the end-to-end lag of the Polhemus, Ascension, and optical ceiling head trackers, while they were in use with Pixel-Planes 5, our custom graphics engine. The measurement scheme is an independent test, relying on electronics and hardware that are completely separate from the tracker and graphics engine themselves.
- We fabricated prototype Self-Tracker sensors through MOSIS.
- We wrote software simulation for Self-Tracker chips using Pixel-Planes 5.

2.4 Interactive Building Walkthrough

- We have increased the degree of user interaction with the model, adding collision detection, the ability to move furniture objects (such as frying pans and chairs), and the ability to "open" doors.
- We have continued running user studies with Walkthrough under the ceiling tracker to study the effectiveness and shortcomings of the existing system.
- We have added real-time model mesh-generation to the Virtus WalkThrough-to-Pixel-Planes 5 modeling pipeline as a preliminary step to adding interactive radiosity.

2.5 Virtual-Environment Ultrasound Scanning System

- Our experimental virtual environment ultrasound scanning system was used by an ultrasound technician from the UNC Hospitals Department of Obstetrics and Gynecology to scan a pregnant woman's fetus (See Appendix B for paper on this topic).

2.6 The Nanomanipulator: An Atomic-scale Teleoperator

- A UCLA-designed scanning tunneling microscope (STM) was installed in the UNC Department of Computer Science graphics and image laboratory, and its operation has been integrated with our head-mounted display and force-feedback manipulator.
- We installed a drive motor on the micrometer for positioning the experimental sample with both manual and computer control. This greatly speeds mounting new samples and bringing them into tunneling range.
- We scanned sample surfaces down to atomic resolution and displayed 3D pictures of these using a head-mounted display.
- We have achieved real-time display of surfaces being scanned by the STM.
- We achieved haptic display of a sample surface using the force-feedback manipulator so that the user can "feel" the surface and can control the position of the probe.

3.0 Expected Milestones during the Next 12 Months

3.1 Head-Mounted Display System

- Begin to add volume-rendering capability to Vlib (which is polygonally based) on Pixel-Planes 5.
- Integrate ultrasound input more uniformly into a virtual world.
- Conduct basic perceptual experiments on what sorts of virtual objects can be successfully superimposed on the real world with a see-through HMD.
- Experiment with new types of hand-held input devices.
- Develop a multiscreen "bathysphere" and integrate it with Vlib.
- Perform a user study to explore the advantages of head-tracked steering in targeting of radiation therapy treatment beams.
- Calculate tracker delay compensation using predictive algorithms for the HMD system.
- Be able to explore and modify molecular surfaces in real time with force feedback and an HMD.
- Be able to switch between different video and serial sources under software control.
- Analyze errors and limitations of 30-degree field-of-view see-through HMD system.
- For the radiation planning application, conduct experiment to investigate relative merits of using natural, intuitive steering, which makes use of proprioceptive and vestibular information, in the exploration of a patient's anatomy and the subsequent targeting of rediotherapy beams.

- For the virtual-environment ultrasound-scanning system:
  - Data acquisition system for 30 Hz ultrasound data update rate in the virtual environment. (Current system updates ultrasound data at 2 Hz).
  - Improved calibration techniques for merging computer generated and real-world images in a virtual environment.
  - Binocular Video See-Through HMD system. (Current system is monocular only).
  - Better depth cues for visualizing ultrasound data within a patient.
  - Reevaluation of improved system with a pregnant subject and ultrasound technician.

3.2 See-through HMD with Custom Optics

- Complete the calibration procedure for the see-through HMD system.
- Complete the first set of perceptual experiments that study the relationships of perceived depths and sizes of virtual and real objects occupying both same or different spaces.

3.3 Tracking

- Develop and demonstrate a method for measuring accurately the locations of all the LEDs in the ceiling, without the use of additional hardware or an unreasonable investment of labor. Such a method would allow the fabrication specifications of some of the ceiling hardware to be relaxed, thereby lowering the costs of manufacture. Such a method may also improve tracking performance if it measures the LED positions more accurately than the precision provided by the careful construction of the present ceiling.
- Continue incremental improvements in the optical tracker. We recently hired a mechanical engineer, who can help us in the following tasks:
  - Designing a new head unit with integral photo diodes and lenses to reduce further the weight and provide greater range of head motion.
  - Increasing the ceiling structure to around 20' by 20' in order to provide much greater range of motion, both quantitatively and psychologically.
  - Investigating the feasibility of making the system wireless. This tracker actually encourages users to walk around a large area, running the risk of tripping over the supporting cables.
- Explore ways to reduce further the occasional discontinuities in the ceiling tracker's output.
- Convert the HMD attached to the ceiling tracker into a see-through HMD. The Ultrasound group is seeking a more accurate tracker to replace the Polhemus system they currently use, and they have decided to try the ceiling tracker. A see-through HMD will also make latency more visible, giving us incentive to explore delay compensation techniques.
- Develop and demonstrate predictive tracking techniques for HMDs. Learn what is required in the system to support such techniques and discover what limitations they have.
- Exploring technologies for unlimited range tracking in unstructured environments. We know of two technologies that have this potential: Self-Tracker, which we are developing, and inertial

technologies. Both are *relative-mode* trackers that measure only the relative differences in position and orientation as the user moves, integrating these differences over time to recover the head's location. The main problem with these technologies is drift, since repeated integration accumulates error over time. Initial systems will probably be hybrids with our optical tracker, using it to provide auxiliary information to evaluate the technology and help it overcome drift by occasionally providing it with a good fix of the true location.

- Continue Self-Tracker development.

  - Identify appropriate personnel to carry the project forward.

  - Design and fabricate operational Self-Tracker sensors.

  - Develop Self-Tracker report processing algorithms.

  - Integrate Self-Tracker chips with board-level interface.

3.4 Interactive Building Walkthrough

- Integrate more advanced versions of Virtus WalkThrough into the system, with support for textures, model partitioning, more complex radiosity emitters, and the replacement of model parts with objects from our model libraries.
- Add real-time, interactive radiosity to the display program on Pixel-Planes 5.
- Move the real-time model mesh-generation to the graphics processors on Pixel-Planes 5 to allow mesh refinement.
- Conduct additional user studies.

3.5 Virtual-Environment Ultrasound Scanning System

- Build a data acquisition system for 30 Hz ultrasound data update rate in the virtual environment (current system updates ultrasound data at 2 Hz).
- Improve calibration techniques for merging computer-generated and real-world images in a virtual environment.
- Begin using binocular video see-through HMD system (current system is monocular only).
- Conduct experimental trials of improved system with an Ob-Gyn physician, an ultrasound technician, and several pregnant subjects.

3.6 The Nanomanipulator

- Increase sampling rate with the STM by two or three orders of magnitude from the current 100 samples per second.
- Modify the STM so that pulses of a few nanoseconds or longer and of a few volts' magnitude can be applied between the probe and the sample.
- Modify the STM so that the wave form of the tunneling current resulting from voltage pulses can be observed and recorded.

- Use correlation of successive scans to compensate for thermal drift and generate a stable image of the sample surface.
- Use voltage pulses to transfer material between the sample and probe.
- Work with our collaborator in this work, Professor Stanley Williams of the UCLA Department of Chemistry, to conduct other experiments possible with these new facilities.

4.0 Discussion of Research

4.1 Head-Mounted Display System

Hardware

The current HMD system is used by many projects and subprojects in a variety of configurations. The following is an overview of the current system:

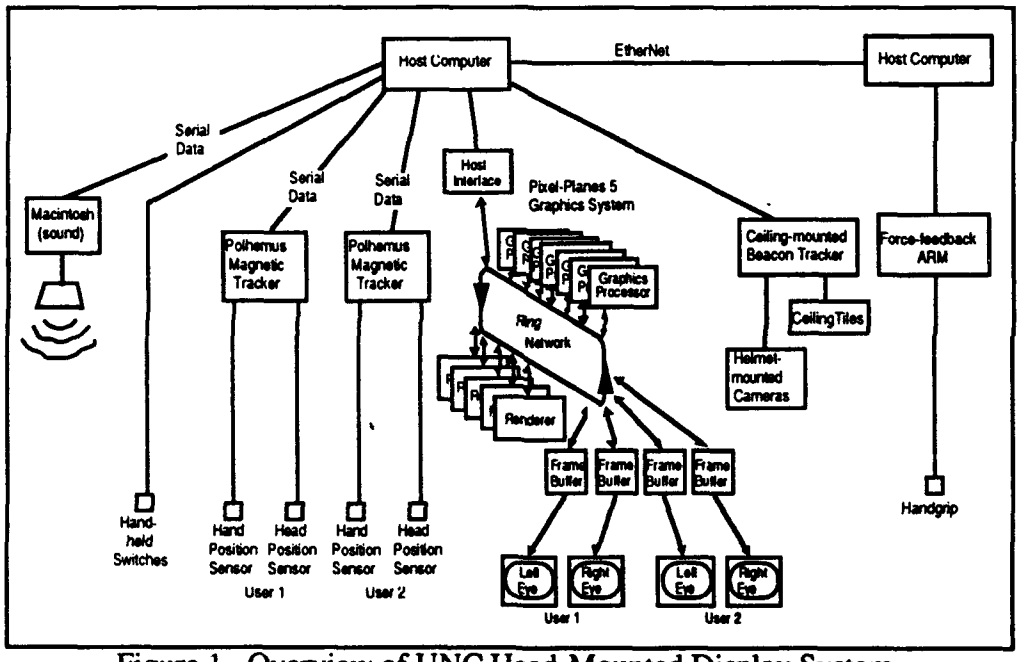


Figure 1. Overview of UNC Head-Mounted Display System.

A description of each component follows:

- Host computer: A Sun 4, which runs the host side of the application and interfaces to the graphics engine.
- Graphics Engine: Pixel-Planes 5, a message-passing multicomputer capable of drawing more than two million Phong-shaded, Z-buffered triangles per second. Multiple applications can run on the system at the same time due to its ability to be split into multiple subsystems.
- Magnetic trackers: We have Fastrak, 3SPACE, and Bird magnetic trackers.

- Ceiling tracker: Described in Section 4.3.
- Force-feedback ARM: Simulates forces in the virtual world; has six degrees of freedom (3 forces and 3 torques); uses computer-controlled servo motors.
- Manual input devices: The newest input device has a natural handgrip shape with five buttons for user input. We also have hollowed-out billiard balls with 3SPACE or Fastrak sensors mounted inside, and switches on the outside for signaling user actions.
- Sound: Controlled by Macintosh; plays sounds under application control either in earphones attached to HMD, or from separate speakers. The sound Macintosh can be controlled by any machine on our network transparently using sdilib.

In general, an application can run with almost any subset of the above equipment.

Software

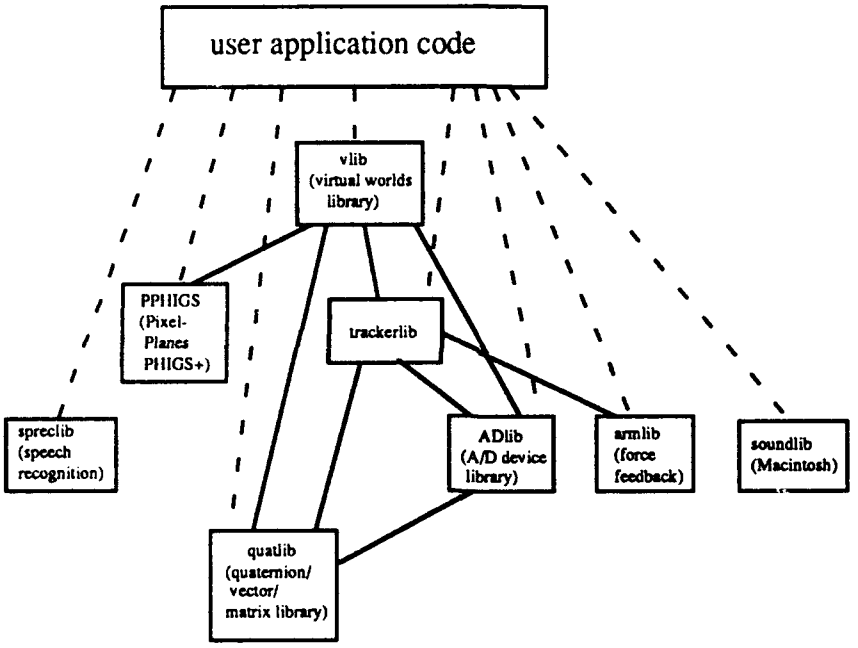


Figure 2. Overview of UNC Virtual-Worlds Software.
(Dashed lines indicate application calls; solid lines indicate inter-library calls.)

The virtual-worlds base software is fairly complete but will continue to expand to be more flexible and powerful and to accommodate new hardware. Most of the libraries support multiple technologies transparently, allowing application code to switch easily between different pieces of hardware by changing only an environment variable. While efforts at improving these libraries will continue, we will work to add new capabilities and develop new applications. The software goals are listed in with the other twelve-month goals for the project in Section 3.1.

Virtual-Environment Ultrasound Scanning System—A video see-through HMD system is used to display ultrasound data in its real-world context. A TV camera is mounted on a conventional HMD to provide video see-through capability. Synthetic images are generated which

correspond to the position and orientation of the TV camera as tracked by the HMD system. The synthetic video images and the live (TV camera) images are then combined in a single image. This creates a virtual environment in which synthetic geometry appears fixed in the real world as the HMD changes position.

The virtual-environment ultrasound scanning system works by gathering ultrasound data and transforming it to match the viewing position of the TV camera mounted on the video see-through HMD. When viewed through the HMD, the transformed ultrasound data appears fixed in its world 3D location within a subject. Two-dimensional ultrasound data is put into our system with a frame grabber and a Polhemus tracker. The frame grabber transfers live 2D images from an ultrasound video output to our image-generation system. The Polhemus tracker is attached to the ultrasound transducer and positions the 2D ultrasound images to their 3D positions in synthetic space.

4.2 See-through HMD with Custom Optics

We are designing a 60-degree field-of-view system which will closely match the geometry of the human head. The human head and shoulder data of one of our project members were collected by a laser scanning device. The geometry of his eyeglasses was measured manually and modeled in Autocad. The optics were then transferred to Autocad, edited to simulate an artificial pupil as well as the CRT that will be used as the display device for the computer-generated images. The models of the head, the eyeglasses, and the optics were all transferred to Pixel-Planes 5 and were visualized in 3D on the high-resolution monitor. The position of the optics was refined using the graphics tool Xfront. The ability to perform such interactive modeling is very helpful to the project since it may avoid the need to build costly mockups.

One of the design modifications we implemented was to fold the miniature CRTs so as to fit the human head more naturally. Despite the fact that such a folding creates some serious redesign efforts, it will result in a more compact and less fragile system. The folding is accomplished by using a folding prism before the CRT as shown in Figure 4.1. We would like to improve the system so that there is enough eye relief to allow HMD users to wear their everyday eyeglasses. This seems to cause more of a feasibility issue with the current design. The wearer of optical glasses will be discussed once the new system with the folding prism is ready to be visualized again in 3D. This new design phase is being carried out with the goal of achieving the same image quality as described in the previous six-month report.

We started working on the design of miniature video cameras to be mounted on a conventional HMD system in order to create a different type of see-through HMD. The main requirement for such a system is to have the two video cameras mounted so that they acquire the same visual scene that the user's eyes would if he/she were not wearing an HMD. For wide field of views of the order of 80 degrees, which currently exist on the conventional HMDs available in our laboratory, such a simulation of the user's eye view is practically non-achievable. The solution to this problem is to redesign another non–see-through HMD to take into account the simultaneous use of two video cameras. This solution is feasible. We have worked out a simple design concept that seems to be promising. In any case, the field-of-view of each video camera that is used to redisplay the acquired video images must match the field-of-view of each of the viewer's eyes. With this in mind, we propose a two-step design process. First, we plan to design new lenses for the video cameras bought off the shelf so that the field-of-views of consideration match. We will then mount the two cameras sideways in front of the viewer with two small folding mirrors. With such an assembly, the inter-pupillary distance of the viewer as well as the line of sight can be matched to the camera positions. The only error left is that the objects in the room will look slightly closer than they would appear with the naked eye, thus creating the impression of looking at the world with a weak magnifying glass. We will be testing this system to see whether or not such a weak mismatch is worth trying to suppress. This might turn out to be an advantage over a perfectly

than they would appear with the naked eye, thus creating the impression of looking at the world with a weak magnifying glass. We will be testing this system to see whether or not such a weak mismatch is worth trying to suppress. This might turn out to be an advantage over a perfectly matching system: this system's slightly magnified view of the objects of interest could prove useful in a medical application in which the physician needs to have a closer view of anatomical structures. If this slight magnification proves unsatisfactory, however, we will need to design a whole new assembly, including the non–see-through viewer itself.

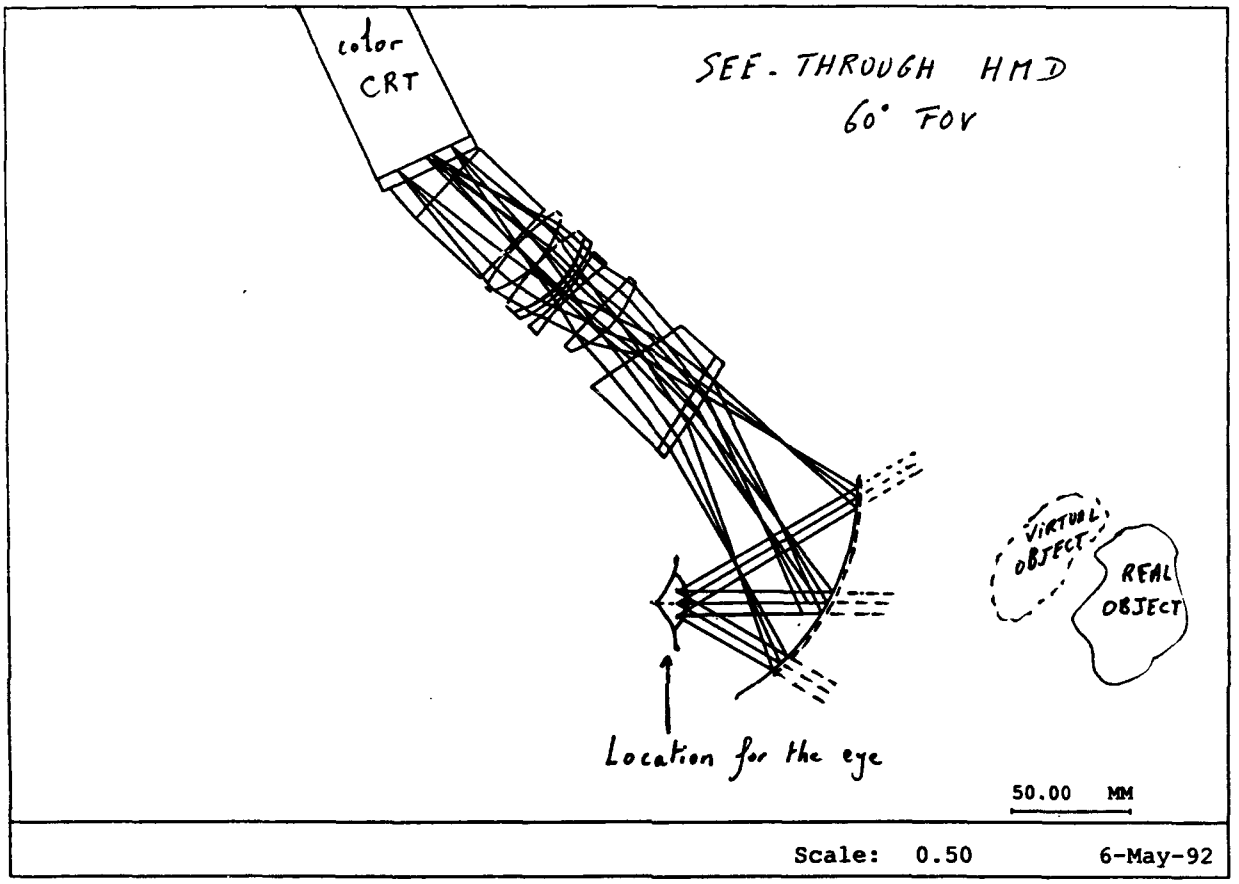


Figure 4.1. Optical Layout of the Folded See-through 60-Degree Field-of-View HMD.

4.3 Tracking

It is well known that trackers for HMDs require high resolution, high update rate, and low latency. Range, however, has not received similar emphasis. Many virtual worlds, such as architectural walkthroughs, would benefit from more freedom of movement than is provided by existing trackers. Long-range trackers would allow larger areas to be explored naturally, on foot, reducing the need to resort to techniques such as "flying" or walking on treadmills. Also, such techniques of extending range work only with closed-view HMDs that completely obscure reality. With see-through HMDs, which we are attempting to build for medical applications, the user's visual connection with reality is intact and hybrid applications are possible where physical objects and computer-generated images coexist. In this situation, flying though the model is meaningless. The model is registered to the physical world and one's relationship to both must change simultaneously.

or inertial systems, that allow unlimited range tracking in unmodified environments. Since such technologies suffer from drift problems, the ceiling tracker will be used as a testbed and an aid in their development.

Ceiling Tracker

The existing full-scale prototype is the second generation of an optoelectronic head-tracking concept that originated at UNC Chapel Hill. The current system places four outward-looking image sensors on the wearer's head and locates LEDs in a 10' x 12' suspended ceiling structure of modular 2' x 2' ceiling panels. Each panel houses 32 LEDs, for a total of 960 LEDs in the ceiling. Images of LEDs are formed by lateral-effect photodiode detectors within each head-mounted sensor. The location of each LED's image on a detector, or *photocoordinate*, is used along with the known LED locations in the ceiling to compute the head's position and orientation. To enhance resolution, the field of view of each sensor is narrow. Thus, each sensor sees only a small number of LEDs at any instant. As the user moves about, the working set of visible LEDs changes, making this a *cellular* head-tracking system. [Ward92]

Measurements of head position and orientation are produced at a rate of 30–150 Hz with 10z–50 ms of delay. Typical values are 80–110 Hz and 15–30 ms of lag, provided that the user stands erect and keeps three or more sensors aimed at the ceiling. The system's accuracy has not been measured precisely, but the short-range resolution has been verified to be within 2 mm and 0.2 degrees.

The system is novel for two reasons. First, the sensor configuration is unique. Other optical tracking systems fix the sensors in the environment and mount the LEDs on the moving body. The outward-looking configuration is superior because it improves the system's ability to detect head rotation. The scalable work space is the system's second contribution. If a larger work space is desired, more panels can be added to the overhead grid.

We discovered that the major problem with cellular tracking systems is that any errors in the system can cause discontinuous jumps in position and orientation as the working set of sources and sensors change. We call this effect beac*on switching error*. We reduced these errors to tolerable levels by precisely placing the beacons and sensors, calibrating systematic distortions in the lens and photodiodes, and using more than the minimum of 3 LEDs to reduce the average error [Azuma91]. Many steps were taken in the low-level software and hardware to maintain high signal-to-noise ratios and achieve good resolution from our sensors.

Another problem with the existing system is the combination of excessive head-born weight and limited rotation range. We would like to add two more sensors to the head unit to provide satisfactory head rotation range, but the head unit already weighs about 8.5 lbs. The total weight of the frame, display, and sensors must be significantly reduced to achieve this goal. Jih-Fang Wang proposed optically multiplexing multiple fields of view onto on a single lateral-effect photodiode. It may be easier, however, to design a new head unit with integral photodiodes and lenses. Given that each photodiode is about the size of a quarter, the entire surface of a head unit could be studded with sensors.

Calibration

The optical tracker software relies on an accurate database of the LED locations in the ceiling. A calibration algorithm is being developed, in collaboration with Professor John F. Hughes of Brown University and Professor Al Barr of California Institute of Technology, which will derive an accurate database from an initial estimate of LED locations, and a collection of raw head unit data.

The present database specifies a grid of coplanar LEDs, perfectly aligned in 24 rows of 40 LEDs each. The present ceiling is built so as to match the positions of the physical LEDs to those specified in the database as closely as possible. Therefore, the accuracy of the database is limited to the tolerances of the current construction. If a ceiling is built, however, even with loose tolerances, the positions of the LEDs could be measured afterwards, and these values could be entered into the database. In this case, the accuracy of the database would be dependent upon the effectiveness of the measurements.

Tracker performance is affected primarily by the *accuracy* of the database. Performance is not dependent on whether the physical ceiling is made to match the database, or vice-versa. Matching the database to a constructed ceiling does not require that the ceiling be built with close physical tolerances, but instead requires an accurate method for measuring LED positions after a ceiling has been built.

Two components, the panels and the superstructure, are the major contributors to the current size-dependent costs of this scalable tracking system. Relaxing the physical tolerances lowers construction costs. The present ceiling panels are solidly built with materials that allow precise machining and resist warping. A redesign of the panels would make them lighter and less expensive. Furthermore, if the panels can be made light enough, they can be dropped into the standard 2' x 2' grid of ceiling tiles found in many buildings, including Sitterson Hall, with a minimum of reinforcement to the existing supports. Thus the ceiling superstructure itself may be dispensed with altogether.

The algorithm being developed will refine a set of initial estimates of the LED positions. These initial estimates will be required to be accurate to within some wide tolerance (perhaps 1"). The algorithm uses a relaxation technique to reconcile the LED database with a large collection of raw head-mount data. The database produced by the current algorithm approaches, but does not yet match, the precision achieved by the ceiling's present careful construction. Improving this algorithm is the subject of current research. The results are so encouraging that many users cannot tell the difference between the self-calibration result and when the precise ceiling locations are used.

Independent Latency Measurements

In an effort to characterize the performance of the various tracking technologies currently in use at UNC-CH, independent tests were performed to measure and compare the system delays for the Polhemus 3Space, the Ascension Bird and the UNC optical ceiling tracker. This was part of a larger effort to measure the end-to-end latencies in head-mounted display systems. The term "end-to-end" latency is used here to describe the total time required for the displayed image in an HMD to change in response to the movement of a user's head.

Measurements were performed by installing special hardware to detect the position of the tracking device sensor independent of the position and orientation readings reported by the device itself. This was accomplished by using combinations of light-emitting diodes (LEDs) and photodiodes. The LED was mounted on or near the tracker position sensor such that its movements were tied to the movement of the sensor. Photodiodes were then placed such that they could detect the motion of the sensor. By comparing the time the LED (and thus the tracker sensor) physically passed in front of the photodiode (indicated by a peak in the measured photodiode output) with the time that the tracker eventually reported being in that position, the measurement latencies for the various tracking devices could be determined. Note that this measured latency includes the time for the sensor to detect its position, the time to transmit that information to the host computer, and the time for the host to access and process the raw tracker data.

Initial results have been obtained and a technical report is being written. Average delays (including the components describe above) have been measured for the Ascension Bird of approximately 85 ms. Corresponding delays of 55 ms and 30 ms have been measured for the Polhemus 3Space and the UNC optical ceiling tracker respectively. Pending it becoming operational, additional measurements will be made to characterize the performance of the new Polhemus Fastrak sensor.

Virtual Walk

On 26 March 1992, Dr. Michael Moshell of the University of Central Florida visited us to conduct a simple experiment with the optical ceiling tracker. The basic idea is this: Can we achieve apparently infinite range out of a finite-range tracker? Suppose that a head-tracker's range could encompass a very large room. A user walking in a straight line would eventually hit the limits of the tracker's range. If, on the other hand, we slowly rotate the virtual world about the user's vertical axis, the user thinks he is walking straight but is actually walking in circles and consequently staying inside the tracker's range! Thus it may be possible to get perceptually infinite range out of such a tracker.

No existing tracker has sufficient range for the task, but our ceiling tracker does let users walk around for several feet. This preliminary experiment gave us some idea of how rapidly we can rotate the world around the user's vertical axis before he notices that something weird is happening. We tested three users. Each user walked from one corner of the space defined by the ceiling tracker to the opposite corner in a fairly brisk three seconds. The user repeated this motion ten times. Each time, the world was either rotated to the left at a certain rate, rotated to the right, or not rotated at all. After the motion was completed, the user had to guess what rotation had been applied. The rotation rate started at zero degrees/sec, smoothly accelerated to a constant number of degrees/sec, and then smoothly decelerated back to zero degrees/sec. This avoided discontinuities that might have made it easier to detect when rotation was used.

We found that users had difficulty detecting surprisingly large rotation rates. Rates of 3–4 degrees/sec were hard to detect. Experience inside HMDs also seemed to be a factor. The two UNC users, who had spent many hours inside the ceiling HMD, had a better guess rate than Dr. Moshell, who had never worn an HMD before. Dr. Moshell had a difficult time detecting rotations of 7–8 degrees/sec.

While this was a preliminary experiment, it suggests that an "infinite virtual walk" may indeed be possible with a finite range tracker.

Self-Tracker

The self-tracker [Bishop84] is one possible inside-out tracking system which may replace, or augment, the optoelectronic tracking system. Its goal is eventually to eliminate fixed beacons and perform tracking using features found naturally in the environment. The self-tracker could then be taken to any room, hallway, or even outdoors.

The self-tracker will use a cluster of custom-made VLSI chips which will be mounted on the head-mounted display and look out on the environment in multiple directions. The cluster of chips will report the apparent shift of the image seen by each chip during successive clock cycles. A host computer will collect the information and calculate the distance the user has traveled relative to his previous position. Approximately 1000 frames per second will be required to track the fastest possible movements reported in human motion literature. This system may not work by itself because of accumulated drift. For the initial versions, a small number of beacons may be placed in the environment to allow for error correction.

Each self-tracker chip will contain a one-dimensional array of photo sensors and supporting image processing circuitry. A one-dimensional array will be used so that a sufficiently large number of

pixels can be placed on the chip and so that image correlation can be performed in real time. The correlation between successive frames will be accomplished by converting the image to a digital representation and then successively shifting the position of one frame and reporting the shift which corresponds to the closest match to the other frame. The tracker chips will be mounted in pairs so that stereopsis can be used to measure the distance to the observed surface in the scene. Each tracker chip will report the observed shift in terms of the number of pixels of shift, so the distance value will be required to convert the shift from pixels to a physical measurement. One chip in each pair will transmit the image's digital representation to the other chip. The second chip will then calculate the correlation between the two images in the same manner that it calculated the correlation between subsequent images on the same chip.

The diagram below is a block diagram of a self-tracker chip. A one-dimensional photo sensor array generates an array of analog voltages which represent the incident light on the array. The edge detection block produces a digital representation of the image. Then the image comparison block compares the current image with the image seen during the last time frame. The shift between the two images is reported, and the current frame is stored for comparison with the next frame. Each image is also sent to its stereo partner to calculate the distance to the observed image.

Self-Tracker research has been hindered by the departure from our department of the graduate student who was working on the project. We are seeking a replacement. Dr. Gary Bishop, who originated Self-Tracker research with his 1984 thesis, has joined our faculty and hopes to guide a new student in the continuation of this research.

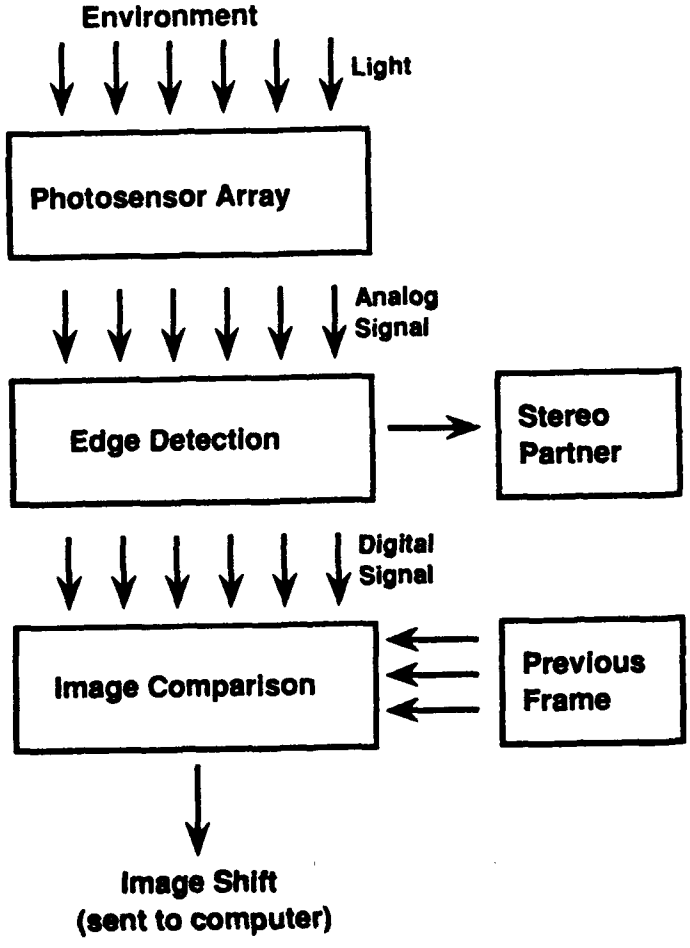


Figure 3. Architecture of the Self-Tracker Custom VLSI Sensor.

Inertial Technologies

Inertial guidance systems, used to steer planes, ships, guided missiles, etc., do not require modification of the environment and have no range restrictions. They could be nearly ideal for tracking HMDs in the future if the endemic drift problems can be controlled. Small, light accelerometers are available today; small, light laser-ring or fiber-optic gyroscopes are just on the threshold of availability to the public and are very expensive. A hybrid approach combining inertial technologies with a system that occasionally provided absolute location fixes to combat drift could be highly effective.

We intend to enter this field by finding and acquiring either rate gyroscopes and/or accelerometers to aid our predictive tracking work. If these prove satisfactory, we will investigate building a stand-alone inertial system.

4.4 Interactive Building Walkthrough

The UNC Interactive Building Walkthrough subproject aims at the development of systems to help architects and their clients explore a building design prior to its construction and, therefore, correct problems on the computer instead of in concrete [Brooks86]. The scenes are shown with radiosity illumination, lights that can be switched on or off with only 100 msec delays, textured surfaces, and near-real-time display from changing viewpoints. Update rates realized by the Pixel-Planes 5 graphics scene generator are 30–40 updates/second (15–20 stereo images/second) on scenes consisting of thousands of polygons, many of which are textured.

Our system uses a head-mounted display and the UNC ceiling tracker to allow the viewer literally to walk freely about a 10' x 12' area of a model. An orange "ribbon" demarcates the bounds of the tracking volume. In larger models, the viewer translates this 10' x 12' area in the direction of gaze with the press of a button. Our research suggests that only translating this area, rather than allowing translations and rotations, may be less disorienting to the viewer.

We have long held that virtual-worlds projects need to move forward simultaneously in four dimensions: faster, prettier, handier, and realer. In the past year, we have made progress in the following areas:

- ***Faster:***

We are taking steps to cut the time between model design iterations. Although modeling will still take place off-line, the other stages—radiosity and partitioning—will be done interactively. We expect interactive radiosity to become available within the next six months and are planning to extract partitioning information from the new version of Virtus WalkThrough.

- ***Prettier:***

We have made texture selection easier: we have created a library of Walkthrough textures and developed tools for making new textures [Rhoades92].

- ***Handier:***

The optical ceiling tracker hardware has proved sufficiently reliable and the freedom of interaction so intuitive that we have retired the treadmill. The tracker team has also added hand tracking to the system, which has made it possible to grab and manipulate objects within the model—a task that would have been difficult with the treadmill, as both hands were required to steer it.

We have experimented with associating volume information with models to support more realistic behavior within the model: chairs may be moved, doors can be opened and shut, and collisions with objects detected. The last, in particular, is important because it makes it possible for us to prevent the disorienting act of walking through a wall.

• ***Realer:***

Uwe Nimscheck, a graduate student visiting from University of Stuttgart, Stuttgart, Germany, wrote a master's thesis here entitled, "Adaptive Mesh Generation for Radiosity Models" [Nimscheck92]. We are now integrating his software into the rest of the run-time environment.

In the past, only the display stage has been interactive, and that interactivity has been limited. Although the lights could be controlled interactively, the geometry was fixed. Any changes in the model had to be sent back through the model pipeline, a process that could take days or weeks. Changes in the model were typically limited to corrections of modeling errors; changes in design were prohibitively time-consuming.

Much of our present activity is devoted to improving the system interactivity, to decreasing the time between model design-cycle iterations. We have begun a collaboration with Virtus Corporation, using their package, Virtus WalkThrough, as a modeling tool. They have provided a version of Virtus WalkThrough from which we can export a model to Pixel-Planes 5 for immediate display. One of our goals is to be able to do interactive design, with a user under the UNC ceiling tracker making design decisions and a modeler using Virtus WalkThrough to carry them out. The user will be able to explore a model, manipulate objects, suggest modifications, and have those modifications implemented in the model around them.

4.5 Virtual-Environment Ultrasound Scanning System

A video see-through HMD system is used to display ultrasound data in its real-world context with the patient on the examining table. A TV camera is mounted on a conventional HMD to provide video see-through capability. Synthetic images are generated which correspond to the position and orientation of the TV camera as tracked by the HMD system. The synthetic video images and the live (TV camera) images are then combined in a single image. This creates a virtual environment in which synthetic geometry appears fixed in the user's "real world" environment as that user, wearing the HMD, moves around.

The virtual-environment ultrasound scanning system works by gathering ultrasound data and transforming it to match the viewing position of the TV camera mounted on the video see-through HMD. When viewed through the HMD, the transformed ultrasound data appears fixed in its world, its 3D location within a subject. Two-dimensional ultrasound data is input into our system with a frame grabber and a Polhemus tracker. The frame grabber transfers live 2D ultrasound video images to our image-generation system, Pixel-Planes 5. A Polhemus tracker is attached to the ultrasound transducer to enable the 2D ultrasound images to be positioned properly in the 3D environment.

This research has been conducted with funds from both the UNC subcontract of an NSF Engineering Research Center grant awarded to Duke University, while utilizing head-mounted display technology being developed under this DARPA contract.

4.6 The Nanomanipulator

We are collaborating with Prof. Stan Williams of the UCLA Department of Chemistry to build a nanomanipulator, consisting of Williams' ultra-low-drift scanning tunneling microscope (STM) driven in real time by our manipulator arm, which the viewer watches real-time reconstruction of the surface, generated on Pixel-Planes 5 and viewed either on a large screen in high resolution or on the head-mounted display. We started this activity in November 1991, got the STM here in Chapel Hill in January 1992, and now have viewing, feeling, and arm-driven STM placement working.

Next comes correlation of successive images so as to measure drift and to be able to compensate for it, and improving the speed of the STM by perhaps 100-fold by new drive electronics, designed by Vern Chi.

Then we will attempt pick-up and release of small amounts of matter by short pulses of voltage about five times the normal imaging voltage. Williams has been working hard on this and believes he can accomplish it.

This work is making use of head-mounted display technology developed under this DARPA contract. Our facilities for haptic display with the force-feedback manipulator were developed with support of NIH. An NSF grant under the Small Grants for Exploratory Research (SGER) program has paid for moving the STM here from UCLA and for the salary of the one graduate student currently working on this project. We have applied to NSF for ongoing support of this project under the HPCC Grand Challenge Applications Program.

References:

[Azuma91] Azuma, Ronald and Mark Ward. Space-resection by collinearity: mathematics behind the optical ceiling head-tracker. University of North Carolina at Chapel Hill Department of Computer Science technical report TR 91-048, November 1991.

[Bishop84] Bishop, Gary and Henry Fuchs. "The Self-Tracker: A Smart Optical Sensor on Silicon," *Proceedings of the 1984 MITT Conference on Advanced Research in VLSI* (Dedham, MA: Artech House, January 1984), pp. 65-73.

[Brooks86] Brooks, Jr., F. P. "Walkthrough - A dynamic Graphics System for Simulating Virtual Buildings," *1986 Workshop on Interactive 3D Graphics*, S. Pizer and F. Crow, eds., University of North Carolina at Chapel Hill, October 1986, pp. 9-22.

[Nimscheck92] Nimscheck, Uwe. "Adaptive Mesh Generation for Radiosity Models." UNC-Chapel Hill Department of Computer Science technical report TR92-014.

[Rhoades92] Rhoades, John, Greg Turk, Andrew Bell, Andrei State, Ulrich Neumann, and Amitabh Varshney. "Real-Time Procedural Textures", to appear in the *ACM Computer Graphics*, Proceedings 1992 Symposium on Interactive 3D Graphics.

[Wang90] Wang, Jih-Fang. "A Real-time Optical 3D Tracker for Head-mounted Display Systems." Ph.D. dissertation, UNC-Chapel Hill, March 1990. UNC Department of Computer Science technical report #TR90-011.

[Ward92] Ward, Mark, Ronald Azuma, Robert Bennett, Stefan Gottschalk, Henry Fuchs. "A Demonstrated Optical Tracker With Scalable Work Area for Head-Mounted Display Systems." *Proceedings of 1992 Symposium on Interactive 3D Graphics--A special issue of Computer Graphics* (Cambridge, MA, 29 March - 1 April 1992), 43-52.

5.0 Dissemination of Research

5.1 Publications

Alspaugh, John. "A Short Guide to AutoCAD Drawing Primitives for 3D Computer Graphics Models and the Walkthrough AutoCAD-to-Polygon Conversion Program." UNC Department of Computer Science technical report TR92-019.

Bajura, Mike, Henry Fuchs, Ryutarou Ohbuchi. "Merging Virtual Reality with the Real World: Seeing Ultrasound Imagery within the Patient." To appear in *Proceedings of SIGGRAPH '92*, August 1992.

Butterworth, Jeff, Andrew Davidson, Stephen Hench, and T. Marc Olano. "3DM: A Three-Dimensional Modeler Using a Head-Mounted Display." *ACM Computer Graphics: Proceedings 1992 Symposium on Interactive 3D Graphics* (Cambridge, Mass., April 1992), 135–138.

Chung, James. "A Comparison of Head-tracked and Non-head-tracked Steering Modes in the Targeting of Radiotherapy Treatment Beams." *ACM Computer Graphics: Proceedings 1992 Symposium on Interactive 3D Graphics* (Cambridge, Mass., April 1992), 193–196.

Holloway, Richard L. "Viper: A Quasi-Real-Time Virtual-Worlds Application." UNC Dept. of Computer Science technical report TR92-004.

Holloway, Richard, Henry Fuchs, and Warren Robinett. "Virtual-Worlds Research at the University of North Carolina at Chapel Hill as of February 1992." To appear *Proc. Computer Graphics International*, Japan, June 1992.

Robinett, Warren, and Richard Holloway. "Implementation of Flying, Scaling, and Grabbing in Virtual Worlds." *ACM Computer Graphics: Proceedings 1992 Symposium on Interactive 3D Graphics* (Cambridge, Mass., April 1992), 189–192.

Robinett, Warren, and Jannick Rolland. "A Computational Model for the Stereoscopic Optics of a Head-Mounted Display." *Presence* 1(1), Jan. 1992, 45–62 (a new refereed journal devoted to virtual reality and telepresence). Also *Proc. SPIE Symposium on Electronic Imaging*, vol. 1457: *Stereoscopic Displays and Applications*, February 1991. See also UNC Dept. of Computer Science technical report TR91-009, Feb. 1991.

Robinett, Warren, Russell Taylor, Vernon Chi, William V. Wright, Frederick P. Brooks, Jr., R. Stanley Williams, Eric Snyder. "The Nanomanipulator: An Atomic-Scale Teleoperator." To appear, ACM SIGGRAPH'92 Course Notes, for the course "Implementation of Immersive Virtual Environments."

Varshney, Amitabh, and J. F. Prins. "An Environment-Projection Approach to Radiosity for Mesh-Connected Computers." *Proceedings of the Third Eurographics Workshop on Rendering*, 271–281.

Ward, Mark, Ronald Azuma, Robert Bennett, Stefan Gottschalk, Henry Fuchs. "A Demonstrated Optical Tracker with Scalable Work Area for Head-Mounted Display Systems." *ACM Computer Graphics: Proceedings of the 1992 Symposium on Interactive 3D Graphics* (Cambridge, Mass., April 1992), 43–52.

5.2 Presentations

By: Ronald Azuma
Topic: A Demonstrated Optical Tracker With Scalable Work Area for Head-Mounted Display Systems
Event: 1992 Symposium on Interactive 3D Graphics
Place: Cambridge, Mass.
Date: 29 March–2 April 1992

By: Jeff Butterworth
Topic: 3DM: A Three-Dimensional Modeler Using a Head-Mounted Display
Event: 1992 Symposium on Interactive 3D Graphics
Place: Cambridge, Mass.
Date: 29 March–2 April 1992

By: James Chung
Topic: A Comparison of Head-tracked and Non-head-tracked Steering Modes in the Targeting of Radiotherapy Treatment Beams
Event: 1992 Symposium on Interactive 3D Graphics
Place: Cambridge, Mass.
Date: 29 March–2 April 1992

By: Henry Fuchs
Topic: Virtual-Worlds Research at the University of North Carolina at Chapel Hill as of February 1992
Event: 1992 Computer Graphics International Conference
Place: Tokyo, Japan
Date: 24 June 1992

By: Henry Fuchs
Topic: Virtual Reality and Medicine: The Lunatic Fringe or the Ultimate Display
Event: Medicine Meets Virtual Reality Conference
Place: San Diego, Calif.
Date: 5 June 1992

By: Henry Fuchs
Topic: Virtual Reality in Medical Imaging: The Lunatic Fringe or the Ultimate Display?
Event: First Seminar for the Center for Biomedical Visualization
Place: Jchns Hopkins University, Baltimore, Md.
Date: 14 May 1992

By: Henry Fuchs
Topic: Virtual-Worlds Research at the University of North Carolina at Chapel Hill
Event: Imagina '92 Conference
Place: Monte Carlo, Monaco
Date: 29 January 1992

By: Warren Robinett
Topic: Implementation of Flying, Scaling, and Grabbing in Virtual Worlds
Event: 1992 Symposium on Interactive 3D Graphics
Place: Cambridge, Mass.
Date: 29 March–2 April 1992

By: Jannick Rolland
Topic: Toward the Blending of Real and Virtual Environments
Event: Media Shock: Visual Fusion/Desktop Revolution—Computer Graphics in the Arts and Sciences '92
Place: New York, NY
Date: 8–10 May 1992

By: Amitabh Varshney
Topic: An Environment-Protection Approach to Radiosity for Mesh-Connected Computers
Event: The Third Eurographics Workshop on Rendering
Place: Bristol, UK
Date: 20 May 1992

5.3 A Partial List of the Visitors who observed Head-mounted Display, Tracker, and Walkthrough demonstrations at UNC-Chapel Hill, January–June 1992

- David Zeltzer, MIT Media Lab
- 12 students from the ACM chapter at UNC-Charlotte
- Alan Huber, Jay Messer, Gary Foley, Environmental Protection Agency
- Billy Ray, President of Southern Bell
- Howard Palmes, VP of Network Planning, Bell South Telecommuncations
- Crew from CNN Science and Technology
- Wayne Rosing, Vice-President, Sun Microsystems Labs
- Pete Hodgson, Member of Parliament, New Zealand
- ~15 participants in the NSF Workshop on Research Directions in Virtual Reality
- Dr. Richard Satava, U.S. Army Medical Corps / Stanford Research Institute
- Prof. Bob Lauterborn, UNC School of Journalism
- 5 from Hughes Research Labs
- Joe Jupin and Jack Tubman, Jet Propulsion Lab
- Norbert Gwosdzik and Dr. Klaus Bruehl, Aachen University of Technology, Germany
- Robert Pool, Science Magazine
- Jeff Malacarne, Hughes Aircraft
- Janice Gaston, Winston-Salem Journal, Winston-Salem, NC
- 20 Computer Science students from Old Dominion University, Norfolk, VA
- George Revesz, Chairman, Dept. of Computer Science, UNC-Charlotte
- Jim Brown and Bob Helms, Research Triangle Institute
- 4 from the UNC Clinical Center for the Study of Development and Learning
- Thomas Wendler, Philips, Hamburg, Germany
- Prof. Klaus Schulten, University of Illinois
- Larry Lee, Hugh Patrick, Tom Palmer, North Carolina Supercomputer Center
- Dr. Kay Wagoner, Glaxo Pharmaceutical Co.
- Prof. Jan Hermans, UNC Biochemistry
- Prof. Alexander Tropsha, UNC School of Pharmacy
- Rich Little and Scott Stevens, Software Engineering Institute, Carnegie Mellon Univ.
- 6 from IBM, Research Triangle Park, NC
- Betty Joyce Nash, Greensboro News and Record, Greensboro, NC
- 12 undergraduate computer science majors from Elon College, Burlington, NC

- 4 from NASA Langley, Hampton, Virginia
- George Kelley and Paul Weisman, Williams Air Force Base
- Bill Dally, MIT
- Harvey Goldberg (producer) and crew from CBS Evening News
- Forrest Baskett and Lee Nicholson, Silicon Graphics
- UNC Board of Governors
- Dr. Richard Pope, University of Leeds School of Medicine, UK
- 6 from Becton Dickinson, Research Triangle Park, NC
- Tony Ting, Glenn Joyce, and Cha-mei Tang, Naval Research Lab
- 5 from Hoechst Celanese Co.
- Frank Hepburn and Mete Bayar, Kaiser Electro-Optics, Inc.

6.0 Appendixes

Appendix A
Holloway, Richard, Henry Fuchs, and Warren Robinett. "Virtual-Worlds Research at the University of North Carolina at Chapel Hill as of February 1992." To appear *Proc. Computer Graphics International*, Japan, June 1992.

Appendix B
Bajura, Mike, Henry Fuchs, Ryutarou Ohbuchi. "Merging Virtual Reality with the Real World: Seeing Ultrasound Imagery within the Patient." To appear in *Proceedings of SIGGRAPH '92*, August 1992.

Appendix C
Holloway, Richard L. "Viper: A Quasi-Real-Time Virtual-Worlds Application." UNC Dept. of Computer Science technical report TR92-004.

Appendix D
Alspaugh, John. "A Short Guide to AutoCAD Drawing Primitives for 3D Computer Graphics Models and the Walkthrough AutoCAD-to-Polygon Conversion Program." UNC Department of Computer Science technical report TR92-019.

Appendix E
Varshney, Amitabh, and J. F. Prins. "An Environment-Projection Approach to Radiosity for Mesh-Connected Computers." *Proceedings of the Third Eurographics Workshop on Rendering*, 271–281.

Appendix F
Robinett, Warren, Russell Taylor, Vernon Chi, William V. Wright, Frederick P. Brooks, Jr., R. Stanley Williams, Eric Snyder. "The Nanomanipulator: An Atomic-Scale Teleoperator." To appear, ACM SIGGRAPH'92 Course Notes, for the course "Implementation of Immersive Virtual Environments."

Virtual-Worlds Research at the University of North Carolina at Chapel Hill as of February 1992

February 1992

Richard Holloway
Henry Fuchs
Warren Robinett

Department of Computer Science
University of North Carolina at Chapel Hill
USA

ABSTRACT

This paper gives a very brief look at the origins of virtual-worlds research, defines the major challenges in this field, and gives an overview of the current state of virtual-worlds research at the University of North Carolina, work that has been going on for over two decades. This paper is an update of [Holloway 91].

Key words: virtual reality, virtual worlds, head-mounted displays, image generation.

1. INTRODUCTION

We trace the origin of virtual-worlds research back to the seminal 1965 IFIP address by Ivan Sutherland, called "The Ultimate Display." In this talk, Sutherland postulated displays so realistic that the viewer would have difficulty differentiating computer-generated (virtual) objects from real ones: "The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal" [Sutherland 65]. Thus, by "virtual world," we mean a system that gives the user a sense of being in an environment other than the one he or she is in, or of simulated objects being in this environment: a system where the user suspends disbelief and has a feeling of presence.

Not content to merely imagine such systems, Sutherland soon designed, built, and demonstrated one such system with a realizable subset of these capabilities [Sutherland 68]. The system consisted of a headband with a pair of small CRTs attached to the end of a large instrumented mechanical arm, through which head position and orientation could be determined. Hand position was sensed via a user-held camera grip suspended at the end of three fishing lines whose length was determined by the number of rotations sensed on each of their reels. This subset of the "ultimate" capabilities—stereo images, head tracking, and hand tracking—remains prevalent even today.

With the notable exception of flight training simulators and military applications, however, research on virtual worlds fell into decline after Sutherland's pioneering work. An indication of this lack of interest is that the major computer graphics textbooks of the 1980s (Newman and Sproull, 2nd edition, 1979; Foley and van Dam, 1982) each devoted only a sentence or two to all of head-mounted displays. Work was quietly proceeding, however, in several places, among them Wright-Patterson Air Force Base [Glines 86], NASA Ames Research Center [Fisher, et al. 86], MIT's Media Lab [Callahan 83], CAE Electronics [CAE Electronics 86], and UNC-Chapel Hill (see bibliography). Worthy of special mention is the work done in the area of flight simulators [Schachter 83], which has produced some of the most convincing virtual worlds to date.

The mid-1980s saw a blossoming of activity spawned by the technological advances in small LCD television screens, image-generation systems, and magnetic tracking systems. These advances brought the cost of the components of a virtual-worlds system down to an affordable level and the system performance up to the point that one could hope to get real work done.

Media interest in this area has steadily mounted in recent years, and such coverage may give the impression that this technology is mature and that the major technical problems have been solved. Unfortunately, the major technical problems that Sutherland identified in the late 1960s remain with us today:

1. Image generation to each eye has to be achieved at real-time rates.
2. Tracking of the head and hand has to be determined in real time and with considerable accuracy.
3. Head-gear display devices have to produce high-resolution images and wide-angle views for two eyes.
4. Force feedback to simulate even simple objects is very difficult.

This paper surveys the current solutions at UNC-Chapel Hill to these major problems and to related other ones.

2. VIRTUAL-WORLDS RESEARCH AT UNC-CH IN FEBRUARY 1992

The Department of Computer Science at the University of North Carolina at Chapel Hill has been active in virtual-worlds research for more than two decades. Our research is aimed squarely at solving the technical problems listed above. This section covers all relevant projects and subprojects that touch on the field of what we call "virtual-worlds research." In addition, this section also describes some subprojects (such as audio systems and input devices) which are not major research efforts at UNC but help in making a complete system.

This paper does not cover all of the virtual-worlds research that has been done at UNC, but a fairly complete bibliography of UNC virtual-worlds research appears at the end of this paper for the interested reader. Some previous work is mentioned to give context for current research.

2.1 Technology Subsystems: Hardware

2.1.1 Image Generation: Pixel-Planes

As discussed previously, one of the hardest problems in this field is real-time image generation. The Pixel-Planes project started in 1980 with the idea of using a massively parallel architecture for rendering of complex, polygonally-based models in real time. Since then, the project team has designed and built five generations of custom IC designs, two small prototypes (Pixel-Planes 2 and 3) and two full-sized systems (Pixel-Planes 4 and 5), which are still in use.

Pixel-Planes 5 is a message-passing multicomputer (Figure 1) which divides the image-generation task among approximately forty Intel i860-based graphics processors (for geometric transformations and other calculations) and approximately twenty-five renderers (massively parallel arrays of tiny processors based on custom chips that perform most pixel-oriented calculations). Pixel-Planes 5 can render more than two million Phong-shaded, z-buffered polygons per second at 1280x1024 resolution. It was first used with the head-mounted display (HMD) system in early 1991, and has substantially enhanced the virtual-worlds applications at UNC in image realism, complexity, and speed.

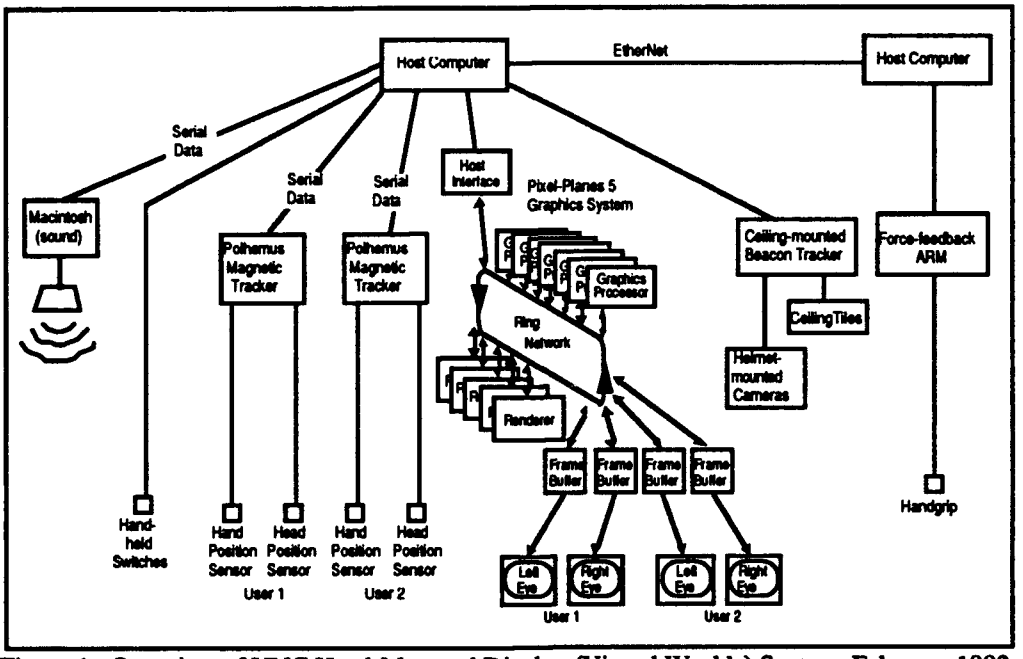


Figure 1. Overview of UNC Head-Mounted Display (Virtual Worlds) System, February 1992.

2.1.2 Tracking

Researchers now at UNC have been investigating the tracking problem for decades [Fuchs, et al. 77; Bishop and Fuchs 84]. Our current tracking systems include both commercial systems and systems developed at UNC. A description of each follows.

Commercial Systems

In 1985, we bought a Polhemus 3SPACE magnetic tracker for use with the first see-through HMD. Since then we have acquired two additional units, plus an Ascension Bird magnetic tracker. These magnetic systems currently have uncomfortably long latency,[1] which limits the real-time system response, and limited range, which constrains the user to a working volume of only a few cubic meters.

The UNC Ceiling Tracker

In an attempt to address the limitations of current magnetic systems, Jih-Fang Wang [Wang 90] designed an optical tracking system using an "inside-out" paradigm pioneered by Bishop [Bishop 84]. In this system, cameras are mounted on the user's head and monitor the relative positions and orientations of infrared LED beacons mounted in the ceiling. From information about three or more beacons, the position and orientation of the cameras (and thus the user's head) can be derived.

Wang designed and constructed a benchtop prototype system which was finished in March 1990. The system was largely reworked by our tracker research group in building a full-scale ceiling tracker, which was completed in July 1991 and demonstrated at the SIGGRAPH '91 conference. The ceiling uses two-foot-square tiles that, in principle, can replace ordinary ceiling tiles in a suspended ceiling framework. The ceiling area is 10 feet by 12 feet at present, which is a considerably larger working volume than that of any trackers known to us. The update rate varies, depending on how many LEDs are visible, but the maximum

[1] The delay between the time the user moves and the time that movement is reported by the tracker.

rate is currently around 100 frames per second, with two frames of latency. Work to improve the system continues, including combining the ceiling tracking system with a magnetic tracker for tracking the user's hand.

2.1.3 Head-Mounted Displays

As with the tracking systems cited above, we use both commercially available systems and "home-grown" systems for our head-mounted displays (HMDs). UNC's work in this area dates back to the early eighties (see history in [Chung et al 89]). Generally speaking, head-mounted displays can be divided into two categories: "see-through" (the computer image is superimposed onto the real world) and opaque (the real world view is blocked by the apparatus). Each type has its own advantages, depending on the application for which it is used.

Commercial Systems

Currently, the HMDs that we use most are made by VPL Research—we have two regular VPL EyePhones (Model 2) plus another EyePhone Model 2 that was modified to hold the special cameras for the ceiling tracker. All of these models are opaque and use color LCD screens and LEEP wide-angle optics.

UNC See-Through HMDs

We are particularly interested in see-through HMDs because they allow computer-generated images to be superimposed on the real world. In this way, the system enhances reality rather than supplants it. An example of this is the "X-ray vision" application described in Section 2.3.4.

Because there were no reasonably priced, commercially available HMDs in 1986, we built our own simple, black-and-white, see-through unit using commercially available LCD screens and half-silvered mirrors. A color version followed in 1987. These models were usable, but suffered from poor image quality and lack of durability.

All of the LCD-based HMDs that we have used suffer from the problem of poor resolution, which leads to grainy, pixelated images[2]. At UNC, Warren Robinett and Jannick Rolland carried out an informal experiment using a virtual Snellen eye chart which showed that a user wearing a VPL EyePhone Model 1 is legally blind, having approximately 20/250 vision. To address this problem, and because we are interested in see-through capability, we are currently building a new see-through HMD using tiny CRTs[3] instead of LCDs. The system under construction will have custom-designed optics with a 60-degree field of view for each eye.

2.1.4 Force Feedback

Research into force feedback at UNC dates back to 1967 when the GROPE I project work used a 2D force-feedback device to enhance understanding of continuous 2D force fields [Batter and Brooks 71].

The main hardware component of subsequent work has been a 6D force-feedback arm donated to us by Argonne National Laboratories. The Argonne Remote Manipulator (ARM) can output three forces and three torques at the handgrip where the user holds it and has a working volume of about one cubic meter. This allows it to simulate a wide variety of virtual forces over a fairly large working volume. This capability has been explored by three different studies: grasping and manipulating of virtual blocks on and over a virtual table [Kilpatrick 76], finding the minimum-energy position and orientation of a virtual bar suspended by springs [Ouh-Young 90], and finding a minimum energy docking configuration for drugs in

---

[2] The resolution of the EyePhone Model 2 is listed as 360x240 in primary-color pixels; this gives something like 208x139 resolution for RGB triads. VPL's EyePhone HRX has approximately 416x277 RGB-triad resolution (the VPL literature lists it as 720x480 primary colored pixels), which is still lower resolution than a standard NTSC video signal, and much less than a 1280x1024 workstation screen.

[3] Current CRT technology affords 1000x1000 resolution for a one-inch-square display.

the active site of a protein molecule [Ouh-Young, Beard and Brooks 90]. A summary of this work is given in [Brooks, et al. 90].

A much simpler force feedback system was implemented for the virtual mountain bike application. Here, a CompuTrainer eddy-current brake is connected to the rear wheel of the mountain bike (described in Section 2.1.6) and controlled by the host computer via a serial line to give force feedback to the rider. When the application detects that the user is going up a hill, it sets the CompuTrainer to a higher braking value to make pedaling more difficult.

2.1.5 Audio

Audio systems have long been in use at UNC for both output (i.e., computer-controlled sounds) and input (i.e., speech recognition). While not a major research effort at UNC, the audio systems have been a useful addition to the virtual-worlds systems developed here.

Audio Output: Kirkpatrick's force-feedback work with virtual blocks used an audible click to signal contact between virtual objects. We now have a dedicated Macintosh IIci sound server for many virtual worlds applications. The Macintosh is connected to the host via a serial line and plays prerecorded sounds under application control. In addition, a CD-ROM player with an extensive sound effects library is connected to the Mac, giving application programs more than 1,000 sounds from which to choose.

Audio Input: In the mid-1980s, the GRIP project used a Votan speech recognizer with the GRINCH system. More recently, we have acquired a DragonWriter speech recognition system for our lab which has been used with some applications. The accuracy of speech recognition systems is still a significant limitation for multi-user systems.

2.1.6 Input Devices

One of the most interesting parts of virtual-worlds applications is the equipment used to get user input to the system. Like the audio subsystems, this is not a major research effort, but it is a necessary part of a complete system. A list of the most important devices follows.

Manual Input Devices

Pool Ball—The most common input device in our lab is currently a hollowed-out billiard ball with a Polhemus sensor placed inside it and two buttons on the outside. This device has served us well since 1986 [Brooks 86] and lends itself well to all sorts of applications. We find it easier to use and more precise than the VPL DataGlove (described below), which is used at many other sites as the primary input device. The advantages of the pool ball are that it does not require calibration for each user, its position can be accurately specified at all times, and actions initiated with the buttons are easy to control precisely.

Finger Pick—A variation on the pool ball, this device consists of a guitar finger pick on which is mounted a Polhemus sensor and a small microswitch. It is simple, intuitive for grabbing, and does not have to be held.

Joybox—An old favorite in our lab for analog input is the joybox, which is a pair of three-degree-of-freedom joysticks with a slider.

DataGlove—Our VPL DataGlove has been used for some applications, but in general, it has more degrees of freedom than most applications need or want. The glove gives information about the angle for all of the user's hand joints, which the application usually has to compress into some small number of *gestures*. Our experience has been that it is usually simpler to encode the gestures using the buttons on the pool ball.

Navigation Devices

Steerable Treadmill—This device addresses the need to virtually walk through computer-simulated environments that are larger than the range of the tracker. A modified treadmill measures how far the user has walked on it, and that information is reflected in the virtual world. A set of bicycle handlebars mounted

on the front allows the user to change direction, and shaft encoders detect the rate of rotation of both the handlebars and the treadmill itself.

Bicycle—Another means of exploring a large virtual world is with a mountain bike. This device consists of a regular mountain bike mounted on a stand, with shaft encoders to detect how fast the rear wheel is spinning and what the current handlebar direction is. As described above, an eddy-current brake on the rear wheel provides force feedback to enhance the realism of this simulation.

2.2 Technology Subsystems: Software

The guiding principle behind the software effort for all of these projects has been device independence. The devices described above are being modified constantly as technology improves; thus there must be a layer of software between the drivers and the applications to shield the applications from low-level changes in the drivers. To that end, a suite of libraries in C has been developed for applications programmers. Generally speaking, application programmers can think in terms of abstract notions such as *displays*, *trackers*, and *input devices*, rather than EyePhone Model 1 Polhemus 3SPACE, etc. Application programs can choose at run-time which display, tracker, and input device they wish to use, simply by changing the value of a UNIX environment variable. The libraries handle the interface so that switching between devices is easy, and changes to the devices themselves do not affect the application code. An interconnection diagram and a brief description of the libraries follows.

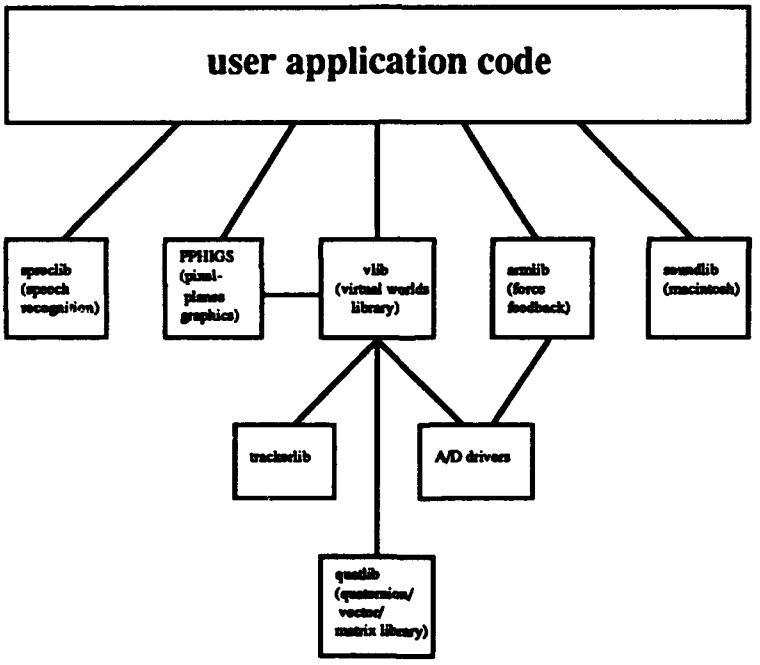


Figure 2. Overview of UNC Virtual-Worlds Software.

Vlib provides a software layer that makes all of the UNC HMDs look logically the same so that applications can switch among them easily. It also provides support for coordinate systems, transformations, and routines for doing common virtual-world operations, such as flying, grabbing, and scaling [Robinett and Holloway 91].

PPHIGS is a local version of the PHIGS+ graphics standard. Applications call both Vlib and PPHIGS in order to build virtual worlds and interact with them.

Trackerlib provides a uniform software layer over all of our current trackers so that applications can access all of them interchangeably.

Armlib hides the low-level details of the ARM's driver and also allows applications to use the ARM from remotely connected machines.

Soundlib is the software library that runs on the UNIX host and communicates with the Macintosh sound server software.

Spreclib is the speech-recognition library and runs on a dedicated IBM-PC speech-recognition server.

ADlib provides a high-level interface to applications to shield them from specific dependencies on input devices.

2.3 Applications

While the pursuit of virtual-worlds technology is interesting and challenging in and of itself, the emphasis at UNC is on improving the technology by tackling real-world applications and letting their problems drive the research. Our belief is that working on real-world applications forces us to solve all the problems in a system, not just the easy ones.

What follows is a short description of the major virtual-worlds applications in progress at UNC.

2.3.1 GRIP: Molecular Studies

The GRIP project started in 1968 with the goal of advancing research in computer graphics by building tools for biochemists. The subprojects of GRIP related to this paper are:

GROPE I, II, and III—Force feedback research described in Section 2.1.4.

Flying Through Molecules—Biochemists can load in a protein molecule modeled as a network of rods or a cluster of spheres, put on the head-mounted display, and enter a world where angstroms are as long as meters, and atoms are as big as beach balls. The user can grab the molecule, scale it up and down, and fly through it.

Trailblazer—This new system uses a large mirror and a CRT with stereo glasses to make a molecular model seem to float in space near the biochemist's hand. The biochemist can reach out using the finger-pick device (described in Section 2.1.6) to grab a virtual molecular bond and move it around. The rest of the molecule flexes to compensate for the changes induced by the user according to physical laws.

2.3.2 Nanomanipulator: Teleoperation at Atomic Scale

We have interfaced to our HMD system a Scanning-Tunneling Microscope (STM) built by our collaborators at the UCLA Department of Chemistry. The STM moves a tiny probe in a raster scan over a sample to collect an array of altitudes, which constitute a three-dimensional surface. This surface can be seen from various viewpoints through the HMD, with the model of the surface being updated as the STM completes successive scans. We have achieved atomic resolution viewing a graphite sample, seeing bumps in the surface in a hexagonal pattern, with each bump being an atom in the graphite's crystal lattice.

Using the HMD to view the data from the STM may be thought of as telepresence at atomic scale, that is, giving the user the sense of presence in the three-dimensional microscopic environment being scanned by the STM. We also have the ability to interpret the surface scanned by the STM as a force field, and use the force-feedback ARM to probe the surface manually and feel its hills and valleys. We believe this new display method for STM data will be useful for chemists as they examine a variety of materials.

2.3.3 Walkthrough

The Walkthrough project began in 1985 with the basic goal of "a virtual building environment, a system which simulates human experience with a building, without physically constructing the building" [Airey, Rohlf and Brooks 90]. Using this system, the user can "walk through" in real time a building that may not

yet exist. As the user moves through the three-dimensional building model, perspective views generated using the radiosity lighting model make it seem as if the user is really inside the building.

The user can view the simulated building with a regular CRT, the large rear-projection screen, or a head-mounted display. The user can move through the building by walking under the ceiling tracker or on the treadmill, or by steering with a joybox.

This system was used to walk through the UNC computer science building before it was built, and, consequently, some design modifications were made before the construction began.

2.3.4 Radiation Treatment Planning

There is currently a large research effort at UNC in employing computer graphics techniques in the area of radiation oncology [Rosenman, et al. 89]. In addition, James Chung is working on a CAD tool using a head-mounted display for designing radiotherapy treatment beam configurations in the hope that it will aid radiotherapists in making better treatment plans.

In this system, a model of the patient's anatomy is explored by a physician who is preparing a radiation therapy treatment plan in order to deliver a lethal dose of radiation to a tumor, while minimizing the exposure of healthy tissue to the radiation. The physician puts on a head-mounted display and walks around a model of the patient's anatomy (obtained by standard computed tomography [CT] methods and rendered as a set of polygons) and some number of polygonally defined radiation beams. The physician experiments with different beam placements by "grabbing" the virtual beam with the 3D mouse in order to find an acceptable placement of the beams as described above.

The problem that this system addresses is that currently radiation treatment planners must look at the patient's anatomy on a two-dimensional screen, which makes it difficult to understand views other than the "cardinal" (orthogonal) views. Treatment geometries involving odd angles, therefore, are not often used even though they might result in a better overall treatment. The projected advantage of this system is that it should allow better treatment plans in less time because in the virtual world the physician is free to examine all angles of beam placement in a natural manner.

2.3.5 X-Ray Vision

X-Ray Vision is a planned head-mounted display application in which a specially designed see-through HMD superimposes the view of the computer-generated world onto the real world.

The images to be superimposed can come from CT, magnetic resonance (MR), ultrasound, or any other imaging technique. Richard Holloway's research involves superimposing static 3D reconstructions of CT or MR images (as opposed to scanning in real-time) onto a real patient. This would allow the surgeon planning reconstructive surgery to see the real soft tissue, yet have a three-dimensional image of the underlying bone at the same time.

2.3.6 See-Through Ultrasound

We have recently begun experimenting with the visualization of real-time clinical ultrasound data within a patient. The visualization is achieved using a small video camera mounted in front of a conventional head-mounted display worn by an observer [Figure 3]. The camera's video images are composited with the computer-generated ones containing one or more 2D ultrasound images properly transformed to the observer's current viewing position [Figure 4]. As the observer walks around the subject, the ultrasound images appear stationary in 3-space within the subject.

This kind of enhancement of the observer's vision may have many other applications, both in medicine—seeing arteries to avoid that lie near a pending surgical incision—and in other areas—seeing inside burning buildings with hand-held radar or walking inside house designs on-site [Bajura 92].

Figure 3. A subject is scanned by an ultrasound technician, while another person looks on with the video see-through head-mounted display. Note the miniature video camera attached above the goggles.

Figure 4. One of the video images presented to the video see-through head-mounted display, here showing several 2D slices in 3D space within the patient's abdomen.

2.3.7 Non-Research Applications

In addition to applications that are designed to solve real-world problems, many smaller projects have been done in order to explore the possibilities of virtual worlds. Many of the following projects came from the biennial course "Exploring Virtual Worlds" at UNC. The lessons learned in the creation of these applications have benefited the research applications as well. A brief summary of a few of these projects follows.

Adventure Game—In a network of rooms connected by portals, the user can see himself in a virtual mirror, use a "vortex gun" to shoot a giant virtual bird, and take a ride in a virtual elevator. *Credits: Warren Robinett.*

Carnival —This application features an amusement park including a train, a ferris wheel, a carousel, and an elevator. *Credits: Erik Erikson.*

City and Lake Flythrough—The user can fly between buildings in a virtual city and through targets that give directions on where to go next. Collisions are detected, and audio and force feedback make the user aware of the collision. In the lake, the user swims with fish that are animated in such a way that they avoid collisions with each other and objects in the lake. *Credits: Ron Azuma, Ulrich Neumann.*

Mountain Bike—The user can ride through a countryside complete with trees, traffic lights, barn, water tower, animated birds, road signs, and pylons that he can run over. *Credits: Erik Erikson, Ryutarou Ohbuchi, Russell Taylor.*

Solar System—Our solar system is modeled to scale; the user can fly between planets and moons, change his frame of reference, orbit on a different planet, and scale the world up and down. *Credits: Warren Robinett.*

3D Modeler—Like MacDraw in 3D. The user can create his own geometric models interactively, out of primitives such as blocks, spheroids, and extruded curves. The system allows two independent users: a designer and an observer/client. *Credits: Drew Davidson, Jeff Butterworth, Marc Olano, Steve Hench.*

Virtual Dog—As a test of the speech-recognition subsystem, a virtual dog was created to respond to user commands. The virtual dog can speak, come, jump, and roll over. *Credits: Bill Brown.*

Virtual Piano—When the user puts on the VPL DataGlove and presses a key on the virtual eleven-key keyboard with his virtual hand, the key descends and a note corresponding to the key that was pressed is played by the Macintosh. *Credits: Bill Brown.*

Virtual Pilot—A minimalist flight simulator. The user flies over a textured landscape of land or sea and controls the flight direction by accelerating or decelerating in the direction he is looking. The land simulation includes an independently flying 747, and the sea environment has an aircraft carrier. *Credits: Trey Greer.*

Virtual Golf—The user is armed with a special putter (whose movement is tracked by the system) in order to sink a virtual golf ball into a virtual cup. When the user sinks the putt, the flag sticking out of the hole turns red and the user hears the sound of a ball falling into a cup. *Credits: Curtis Hill*

3. CONCLUSION

These are times of great change in virtual-worlds research. This is a far cry from the situation of even five years ago, when much of a system had to be custom-built. The technology is sufficiently well-developed now that commercial offerings are available for all of the system components, making it possible for groups without hardware-development capabilities to conduct virtual-worlds research. Although the demands of many applications still exceed the capabilities of the technology, it is encouraging that at least a few applications are ready for end users.

Now, even more difficult and subtle challenges may lie ahead; some of the problems we are currently facing are:

- low image quality of LCDs
- expense and availability of tiny CRT systems
- elimination of the lag between user motion and system response
- wide field of view in stereo that is superimposed on the view of the real world
- superimposition of virtual objects on the real world in a way that makes sense to the human visual system
- comfort vs. encumbrance of virtual-worlds head and body gear
- performance of non-real-time operating systems
- ability to model complex virtual worlds
- image generation for complex scenes

If we cannot overcome these problems, virtual worlds may continue to be limited to a few specialized applications (flight simulation and entertainment, for example). If we *can* solve these problems, then we may look back in twenty years and consider these last few years as the time when human-computer interaction left the confines of the desktop computer screen and merged with the 3D world of the user. If so, today's futuristic virtual-worlds applications will seem to their users itwenty years from now as mundane as today's WYSIWYG "virtual paper" displays appear to today's desk-top publishers.

4. CREDITS

Image Generation: Pixel-Planes
PIs: Henry Fuchs, John Poulton *Software Manager:* Anselmo Lastra
Current: Mike Bajura, Andrew Bell, Jeff Butterworth, David Ellsworth, John Eyles, David Harrison, Edward (Chip) Hill, Vicki Interrante, Jonathan Leech, Steve Molnar, Carl Mueller, Ulrich Neumann, Marc Olano, Madhar Ponamgi, John Rhoades, Greg Turk, Laura Weaver
Former: John Austin, Howard Good, Trey Greer, Mark Parris, Brice Tebbs, Russ Tuck

Tracking
PI: Henry Fuchs
Current: Ron Azuma, Brad Bennett, Gary Bishop, Vern Chi, Stefan Gottschalk, Phil Jacobsen, Mark Mine
Former: Carney Clegg, Jack Kite, Jih-Fang Wang, Mark Ward

Head-Mounted Displays
PIs: Frederick Brooks, Jr., and Henry Fuchs
Current: Gary Bishop, Jim Chung, Drew Davidson, Erik Erikson, David Harrison, Rich Holloway, Doug Holmgren, John Hughes, Steve Pizer, Jannick Rolland, Russell Taylor
Former: Mark Harris, Warren Robinett (Project Managers)
Bill Brown, Clement Cheung, Brad Crittenden, Mike Kelley, Ming Ouh-Young, Mike Pique

Force Feedback
(Most of the force-feedback research has been done as part of the GRIP project.)
Current: David Bennett, Jeff Chen, John Hughes, William Wright
Former: James Batter,, Jerome Kilpatrick, Margaret Minsky, Ming Ouh-Young, Mike Pique, Neela Srinivasan

Audio
(Most of the audio research has been done as part of the HMD and Walkthrough projects.)
Current: Drew Davidson, John Hughes, Warren Robinett
Former: Bill Brown, Xialin Yuan

Input Devices
(Most of the input device research has been done as part of the HMD, Walkthrough, and GRIP projects.)
Current: Jim Chung, Drew Davidson, Erik Erikson, David Harrison, Rich Holloway, John Hughes, Warren Robinett, Russell Taylor
Former: Jack Kite, Phil Stancil

GRIP: Molecular Studies
(The following lists only the project members and collaborators whose work touched on a virtual-worlds application; the list of all project members and collaborators is much longer.)
PI: Frederick P. Brooks, Jr.
Current : William Wright, Project Director
Andrew Certain, Jeff Chen, Jim Chung, Steve Hench, Richard Holloway, John Hughes, David Richardson, Jane Richardson, Russell Taylor, Mark Surles
Former : Project Directors: Mark Harris, Mike Pique, Warren Robinett, Helga Thorvaldsdottir
James J. Batter, Bill Brown, Joe Capowski, Jerome Kilpatrick, Ming Ouh-Young

Nanomanipulator
PIs: Warren Robinett, William Wright
PI (UCLA): R. Stanley Williams
Current: Russell Taylor, Vernon L. Chi, Frederick P. Brooks, Jr., Eric J. Snyder (UCLA)

Walkthrough
PI: Frederick P. Brooks, Jr.
Current: John Alspaugh, John Hughes, Yulan Wang, Xialin Yuan
Former: John Airey, Randy Brown, Curtis Hill, John Rohlf, Douglass Turner, Amitabh Varshney, Xialin Yuan

Radiation Treatment Planning
(part of the Head-Mounted Display Project)
Current: James Chung, Stephen Pizer, Julian Rosenman

X-Ray Vision
(part of the Head-Mounted Display Project)
Current: Gary Bishop, Jefferson Davis, Richard Holloway, Doug Holmgren, John Hughes, Warren Robinett, Jannick Rolland

5. ACKNOWLEDGMENTS

The work described in this paper has been supported by one or more of the following grants:

Defense Advanced Research Projects Agency, Contract No. DAEA 18-90-C-0044: "Advanced Technology for Portable Personal Visualization" (Frederick P. Brooks, Jr., and Henry Fuchs, principal investigators)

Digital Equipment Corporation, Research Agreement No. 582: "Interactive 3D Computer Graphics Systems and Applications" (Henry Fuchs, principal investigator)

National Institutes of Health, National Center for Research Resources, Grant No. 5-R24-RR-02170: "Interactive Graphics for Molecular Studies" (Frederick P. Brooks, Jr., principal investigator)

National Institutes of Health, Grant No. P01 CA47982: "Program Project Grant on Medical Image Presentation" (Stephen M. Pizer, program director; especially the "Three Dimensional Medical Image Display," subprogram, Henry Fuchs, principal investigator)

National Science Foundation, Grant No. CDA-8722752: "CISE Institutional Infrastructure Program for Prototyping Complete Digital Systems" (Vernon Chi, Susanna Schwab, and Stephen Weiss, principal investigators)

National Science Foundation, Grant No. MIP-9000894, and Defense Advanced Research Projects Agency, Order No. 7510: "Supercomputing Power for Interactive Visualization" (Henry Fuchs and John Poulton, principal investigators)

National Science Foundation, Cooperative Agreement No. ASC-8920219, and Defense Advanced Research Projects Agency: "Science and Technology Center for Computer Graphics and Scientific Visualization" (Donald Greenberg, center director. Principal investigators at the five participating institutions are: Andries van Dam, Brown University; Alan Barr, California Institute of

Technology; Donald Greenberg, Cornell University; Henry Fuchs, University of North Carolina at Chapel Hill; Richard Riesenfeld, University of Utah).

National Science Foundation, Grant No. CCR 8609588, "Walkthrough—A Dynamic Graphics System for Simulating Virtual Buildings" (Frederick P. Brooks, Jr., principal investigator)

Office of Naval Research, Grant No. N00014-86-K-0680: "The Infrastructure of Command Information Systems" (Stephen Weiss, Frederick P. Brooks, Jr., and Donald Stanat, principal investigators)

6. BIBLIOGRAPHY

6.1 Cited References for Work Done Elsewhere

CAE Electronics, Ltd. 1986. Introducing the visual display system you wear. Quebec. 4pp.

Callahan, M.A. 1983. A 3-D display head-set for personalized computing. MS thesis. Department of Architecture. Massachusetts Institute of Technology.

Fisher, S.S., M. McGreevy, J. Humphries, and W. Robinett. 1986. Virtual environment display system. *ACM Proceedings of the 1986 Workshop on Interactive 3D Graphics*. 77–87.

Fuchs, Henry, Joe Duran and Brian Johnson. 1977. A system for automatic acquisition of three-dimensional data. In *Proceedings of the 1977 National Computer Conference*. 46:49–53.

Glines, C.V. 1986. Brain buckets. *Air Force Magazine*. 69:8:86–90.

Rheingold, Howard. 1991. *Virtual Reality*. New York. Summit.

Schachter, B. 1983. *Computer Image Generation*. New York. Wiley.

Sutherland, Ivan. 1965. The ultimate display. *Information Processing 1965: Proceedings of IFIP Congress 65*. 506–508.

Sutherland, Ivan. 1968. A head-mounted three-dimensional display. *Proceedings of the 1968 Fall Joint Computer Conference*. Thompson Books. 757–764.

6.2 Work Done at UNC

Architectural Walkthrough

Airey, John, John Rohlf, and Frederick P. Brooks, Jr. 1990. Towards image realism with interactive update rates in complex virtual building environments. *Computer Graphics: Proceedings of 1990 Symposium on Interactive 3D Graphics*. Snowbird, Utah. 41–50.

Brooks, Frederick P., Jr. 1986. Walkthrough—A dynamic graphics system for simulating virtual buildings. In *Proceedings of 1986 Workshop on Interactive 3D Graphics*. UNC-Chapel Hill. 9–21.

Force Feedback

Batter, J.J. and F.P. Brooks, Jr. 1971. GROPE-I: A computer display to the sense of feel. In *Information Processing: Proceedings of IFIP Congress 71*. 759–763.

Brooks, F.P., Jr., M. Ouh-Young, J.J. Batter, and P.J. Kilpatrick. 1990. Project GROPE—Haptic displays for scientific visualization. *Computer Graphics: Proceedings of SIGGRAPH '90*. Dallas,TX. 177–185.

Capowski, J.J. 1971. Remote manipulators as a computer input device. MS thesis. University of North Carolina, Chapel Hill.

Kilpatrick, P.J. 1976. The use of kinesthetic supplement in an interactive graphics system. PhD dissertation. University of North Carolina, Chapel Hill.

Minsky, Margaret, Panel Chair. 1989. Recent progress creating environments with the sense of feel: Giving look and feel its missing meaning. *Proceedings of CHI '89*. Austin, TX. F.P. Brooks, Jr., M. Behensky, D. Milliken, M. Russo and A. Druin, panelists.

Minsky, Margaret, Ming Ouh-Young, Oliver Steele, Frederick P. Brooks, Jr., and Max Behensky. 1990. Feeling and seeing: Issues in force display. *Computer Graphics: Proceedings of 1990 Symposium on Interactive 3D Graphics*. Snowbird, Utah. 235–244.

Ouh-Young, Ming. 1990. Force display in molecular docking. PhD dissertation. University of North Carolina, Chapel Hill.

Ouh-Young, Ming, D.V. Beard and F.P. Brooks, Jr. 1989. Force display performs better than visual display in a simple 6-D docking task. *Proceedings of IEEE 1989 Robotics and Automation Conference*. Scottsdale, AZ. 3:1462–1466.

Ouh-Young, Ming, M. Pique, M. Harris, J. Hughes, N. Srinivasan and F.P. Brooks, Jr. 1988. Force display in molecular docking. Abstract in *Journal of Molecular Graphics*. 6:4:224.

Ouh-Young, Ming, M. Pique, J. Hughes, N. Srinivasan and F.P. Brooks, Jr. 1988. Using a manipulator for force display in molecular docking. *Proceedings of IEEE 1988 Robotics and Automation Conference*. Philadelphia, Pennsylvania. 3:1824–1829.

Steelman, H.S. 1968. The GROPE-I system: An analysis of friction and backlash problems. MS thesis. University of North Carolina, Chapel Hill.

Head-Mounted Display

Brooks, Frederick P. Jr. 1988. Grasping reality through illusion. *Proceedings of CHI '88*. Washington, D.C.

Chung, J.C., M.R. Harris, F.P. Brooks, Jr., H. Fuchs, M.T. Kelley, J. Hughes, M. Ouh-Young, C. Cheung, R.L. Holloway and M. Pique. 1989. Exploring virtual worlds with head-mounted displays. *Non-Holographic True 3-Dimensional Display Technologies: SPIE Proceedings*. Los Angeles, CA. 1083:15–20.

Holloway, R.L. 1987. Head-mounted display. UNC technical report.

Holloway, R.L, Henry Fuchs, Warren Robinett. 1991. Virtual-worlds research at the University of North Carolina at Chapel Hill. *Proceedings of Computer Graphics*. London. 181–196. Also printed in *Proceedings of Imagina '92*. Monte Carlo.

Lipscomb, J.S. 1987. Comparison of stereoscopic display devices at the University of North Carolina at Chapel Hill. *Chemical Design Automation News*. 2:5:3–6.

Robinett, Warren, and Jannick Rolland. 1991. A computational model for the stereoscopic optics of a head-mounted display. To appear in *Presence. 1:1*.

Robinett, Warren. 1991. Electronic expansion of human perception. *Whole Earth Review*. 16–21.

Robinett, Warren, and Richard Holloway. 1991. Flying, grabbing and scaling in virtual reality. Submitted for publication.

Medical Imaging

Bajura, Mike, Henry Fuchs, Ryutarou Ohbuchi. 1992. Merging virtual reality with the real world: seeing ultrasound imagery within the patient. UNC Department of Computer Science Technical Report #TR92-005, January 1992.

Levoy, Marc, and Ross Whitaker. 1990. Gaze-directed volume rendering. *Computer Graphics: Proceedings of 1990 Symposium on Interactive 3D Graphics*. Snowbird, Utah. 24:2:217–224.

Mills, Peter H., Henry Fuchs. 1990. 3D ultrasound display using optical tracking. *Proceedings of the First Conference on Visualization in Biomedical Computing*. Atlanta, GA. 490–497.

Ohbuchi, Ryutarou and Henry Fuchs. 1990. Incremental 3D ultrasound imaging from a 2D scanner. *Proceedings of the First Conference on Visualization in Biomedical Computing*. Atlanta, GA. 360–367.

Ohbuchi, Ryutarou, and Henry Fuchs. 1991. Incremental volume rendering algorithm for interactive 3D ultrasound imaging. *Proceedings of the Information Processing in Medical Imaging (IPMI) Conference XII*. 486–500.

Rosenman, Julian, George W. Sherouse, Henry Fuchs, Stephen M. Pizer, Andrew L. Skinner, Charles Mosher, Kevin Novins and Joel E. Tepper. 1989. Three-dimensional display techniques in radiation therapy treatment planning. *Int. J. Radiation Oncology Biol. Phys.* 16:1.

Pixel-Planes

Eyles, J., J. Austin, H. Fuchs, T. Greer and J. Poulton. Pixel-Planes 4: A summary. *Proceedings of the Eurographics '87 Second Workshop on Graphics Hardware: Advances in Computer Graphics Hardware II*. 183–208.

Fuchs, H., J. Goldfeather, J.P. Hultquist, S. Spach, J.D. Austin, F.P. Brooks, Jr., J.G. Eyles and J. Poulton. 1985. Fast spheres, shadows, textures, transparencies, and image enhancements in Pixel-Planes. *Computer Graphics: SIGGRAPH '85 Conference Proceedings*. 19:3:111-120. Reprinted in *Advances in Computer Graphics I*. G. Enderle, M. Grave, and F. Lillehagen, eds. Springer-Verlag. 1986. 169-187. Also reprinted in *Tutorial: Computer Graphics Hardware—Image Generation and Display*. Hassan K. Reghbati and Anson Y.C. Lee, eds. IEEE Computer Society Press. 1988. 222–231.

Fuchs, Henry, John Poulton, John Eyles, Trey Greer, Jack Goldfeather, David Ellsworth, Steve Molnar, Greg Turk, Brice Tebbs and Laura Israel. 1989. A heterogeneous multiprocessor graphics system using processor-enhanced memories. *Computer Graphics: Proceedings of SIGGRAPH '89*. 23:4:79–88.

Tracking

Bishop, Gary. 1984. Self-tracker: A smart optical sensor on silicon. PhD dissertation. University of North Carolina, Chapel Hill.

Bishop, Gary, and Henry Fuchs. 1984. The self-tracker: A smart optical sensor on silicon. *Proceedings of the 1984 MIT Conference on Advanced Research in VLSI*. Dedham, MA. Artech House. 65–73.

Wang, J., R. Azuma, G. Bishop, V. Chi, J. Eyles, and H. Fuchs. 1990. Tracking a head-mounted display in a room-sized environment with head-mounted cameras. *Proceedings: SPIE '90 Technical Symposium on Optical Engineering & Photonics in Aerospace Sensing*. Orlando, FL.

Wang, J., V. Chi and H. Fuchs. 1990. A real-time optical 3D tracker for head-mounted display systems. *Computer Graphics: 1990 Symposium on Interactive 3D Graphics*. 24:2:205–215.

Wang, Jih-Fang. 1990. A real-time optical 6D tracker for head-mounted display systems. PhD dissertation. University of North Carolina, Chapel Hill.

Richard Holloway is a PhD student working on the Head-Mounted Display project, which is the central project for virtual worlds research at UNC. He has written the base software libraries for the project, which are now used by almost all current virtual-worlds applications at UNC. His research project is "X-Ray Vision for Cranio-Facial Reconstruction Planning."

Henry Fuchs is Federico Gil professor of computer science and adjunct professor of radiation oncology at the University of North Carolina at Chapel Hill. He received a BA in information and computer science from the University of California at Santa Cruz in 1970 and a PhD in computer science from the University of Utah in 1975. He serves on advisory committees, including that of NSF's Division of Microelectronic Information Processing Systems and ShoGraphics' Technical Advisory Board. Current research interests include high-performance graphics hardware, 3D medical imaging, and head-mounted display and virtual environments. He and Kenan professor Frederick P. Brooks, Jr., lead UNC's Head-Mounted Display Project.

Warren Robinett is a designer of interactive computer graphics software and hardware. He designed the Atari *Adventure*, the first graphical adventure video game. He was co-founder and chief software engineer at The Learning Company, a publisher of educational software. There he designed *Rocky's Boots*, a computer game which teaches digital logic design to children. *Rocky's Boots* won Software of the Year awards from three magazines in 1983. Robinett worked as a research scientist at NASA Ames Research Center, where he designed the software for the Virtual Environment Workstation, NASA's pioneering virtual reality project. From 1989 to 1991, he managed the Head-Mounted Display project at the University of North Carolina and is now co-director of the Nanomanipulator project at UNC.

Authors can be reached at this address:

Department of Computer Science
Sitterson Hall
University of North Carolina
Chapel Hill, NC 27599-3175 USA
Tel: 919-962-1700 Fax: 919-962-1799

Merging Virtual Objects with the Real World:
Seeing Ultrasound Imagery within the Patient

Michael Bajura, Henry Fuchs, and Ryutarou Ohbuchi

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175

Abstract

We describe initial results which show "live" ultrasound echography data visualized within a pregnant human subject. The visualization is achieved by using a small video camera mounted in front of a conventional head-mounted display worn by an observer. The camera's video images are composited with computer-generated ones that contain one or more 2D ultrasound images properly transformed to the observer's current viewing position. As the observer walks around the subject, the ultrasound images appear stationary in 3-space within the subject. This kind of enhancement of the observer's vision may have many other applications, e.g., image guided surgical procedures and on location 3D interactive architecture preview.

CR Categories: I.3.7 [Three-Dimensional Graphics and Realism] Virtual Reality, I.3.1 [Hardware architecture]: Three-dimensional displays, I.3.6 [Methodology and Techniques]: Interaction techniques, J.3 [Life and Medical Sciences]: Medical information systems.

Additional Keywords and Phrases: Virtual reality, see-through head-mounted display, ultrasound echography, 3D medical imaging

1. Introduction

We have been working toward an 'ultimate' 3D ultrasound system which acquires and displays 3D volume data in real time. Real-time display can be crucial for applications such as cardiac diagnosis which need to detect certain kinetic features. Our 'ultimate' system design requires advances in both 3D volume data *acquisition* and 3D volume data *display*. Our collaborators, Dr. Olaf von Ramm's group at Duke University, are working toward real-time 3D volume data acquisition [Smith 1991; von Ramm 1991]. At UNC-Chapel Hill, we have been conducting research on real-time 3D volume data visualization.

Our research efforts at UNC have been focused in three areas: 1) algorithms for acquiring and rendering real-time ultrasound data, 2) creating a working virtual environment which acquires and displays 3D ultrasound data in real time, and 3) recovering structural information for volume rendering specifically from ultrasound data, which has unique image processing requirements. This third area is presented in [Lin 1991] and is not covered here.

Section 2 of this paper reviews previous work in 3D ultrasound and Section 3 discusses our research on processing, rendering, and displaying echographic data without a head-mounted display. Since the only real-time volume data scanners available today are 2D ultrasound scanners, we try to approximate our 'ultimate' system by incrementally visualizing a 3D volume dataset reconstructed from a never-ending sequence of 2D data slices [Ohbuchi 1990; 1991]. This is difficult because the volume consisting of multiple 2D slices needs to be visualized incrementally as the 2D slices are acquired. This incremental method has been successfully used in off line experiments with a 3-degree-of-freedom (DOF) mechanical arm tracker and is extendible to 6 degrees of freedom, e.g., a 3D translation and a 3D rotation, at greater computational cost.

Sections 4 and 5 present our research on video see-through head-mounted display (HMD) techniques involving the merging of computer generated images with real-world images. Our video see-through HMD system displays ultrasound echography image data in the context of real (3D) objects. This is part of our continuing see-through HMD research, which includes both optical see-through HMD and video see-through HMD. Even though we concentrate here on medical ultrasound imaging, applications of this display technology are not limited to it (see Section 6.2).

2. Previous Research in 3D Ultrasound

The advantages of ultrasound echography are that it is relatively safe compared with other imaging modalities and that images are generated in real time [Wells 1977]. This makes it the preferred imaging technique for fetal examination, cardiac study, and guided surgical procedures such as fine-needle aspiration biopsy of breast tumors [Fornage 1990]. Ultrasound echography offers the best real-time performance in 3D data acquisition, although slower imaging modalities such as MRI are improving.

The drawbacks of ultrasound imaging include a low signal to noise ratio and poor spatial resolution. Ultrasound images exhibit "speckle" which appears as grainy areas in images. Speckle arises from coherent sound interference effects from tissue substructure. Information such as blood flow can be derived from speckle but in

appear in Proceedings of SIGGRAPH '92
hicago, July 26-31, 1992). In Computer
aphics 26, 4 (July 1992).

general speckle is hard to utilize [Thijssen 1990]. Other problems with ultrasound imaging include attenuation that increases with frequency, phase aberration due to tissue inhomogeneity, and reflection and refraction artifacts [Harris 1990] .

2.1 3D Ultrasound Image Acquisition

Just as ultrasound echography has evolved from 1D data acquisition to 2D data acquisition, work is in progress to advance to 3D data acquisition. Dr. Olaf von Ramm's group at Duke University is developing a 3D scanner which will acquire 3D data in real time [Shattuck 1984; Smith 1991; von Ramm 1991]. The 3D scanner uses a 2D phased array transducer to sweep out an imaging volume. A parallel processing technique called *Explososcan* is used on return echoes to boost the data acquisition rate.

Since such a real-time 3D medical ultrasound scanning system is not yet available, prior studies on 3D ultrasound imaging known to the authors have tried to reconstruct 3D data from imaging primitives of a lesser dimension (usually 2D images). To reconstruct a 3D image from images of a lesser dimension, the location and orientation of the imaging primitives must be known. Coordinate values are explicitly tracked either acoustically [Brinkley 1978; King 1990; Moritz 1983], mechanically [Geiser 1982a; Geiser 1982b; Hottier 1989; McCann 1988; Ohbuchi 1990; Raichelen 1986; Stickels 1984], or optically [Mills 1990]. In other systems, a human or a machine makes scans at predetermined locations and/or orientations [Collet Billon 1990; Ghosh 1982; Itoh 1979; Lalouche 1989; Matsumoto 1981; Nakamura 1984; Tomographic Technologies 1991].

A particularly interesting system under development at Philips Paris Research Laboratory is one of the closest yet to a real-time 3D ultrasound scanner [Collet Billon 1990]. It is a follow on to earlier work which featured a manually guided scanner with mechanical tracking [Hottier 1990]. This near real-time 3D scanner is a mechanical sector scanner, in which a conventional 2D sector scanhead with an annular array transducer is rotated by a stepper motor to get a third scanning dimension. In a period of 3 to 5 seconds, 50 to 100 slices of 2D sector scan images are acquired. Currently the annular array transducer in this system provides better spatial resolution, but less temporal resolution, than the real-time 3D phased array system by von Ramm et al., mentioned above. A commercial product, the *Echo-CT* system by Tomographic Technologies, GMBH, uses the linear translation of a transducer inside a tube inserted into the esophagus to acquire parallel slices of the heart. Image acquisition is gated by respiration and an EKG to reduce registration problems [Tomographic Technologies 1991].

2.2 3D Ultrasound Image Display

One should note that 3D image data can be presented not only in visual form, but also as a set of calculated values, e.g., a ventricular volume. The visual form can be classified further by the rendering primitives used, which can be either geometric (e.g., polygons) or image-based (e.g., voxels). Many early studies focused on non-invasively estimating of the volume of the heart chamber [Brinkley 1978; Ghosh 1982; Raichelen 1986; Stickels 1984]. Typically, 2D echography (2DE) images were stored on video tape and manually processed off-line. Since visual presentation was of secondary interest, wire frames or a stack of contours were often used to render geometrical reconstructions.

An interesting extension to 2D display is a system that tracks the location and orientation of 2D image slices with 6 DOF [King 1990]. On each 2D displayed image, the system overlays lines indicating the intersection of the current image with other 2D images already acquired. The authors claim that these lines help the viewer understand the relationship of the 2D image slices in 3D space. Other studies reconstructed 3D grey level images preserving grey scale, which can be crucial to tissue characterization [Collet Billon 1990; Hottier 1989; Lalouche 1989; McCann 1988; Nakamura 1984; Pini 1990; Tomographic Technologies 1991]. [Lalouche 1989] is a mammogram study using a special 2DE scanner that can acquire and store 45 consecutive parallel slices at 1 mm intervals. A volume is reconstructed by cubic-spline interpolation and then volume rendered. [McCann 1988] performed gated acquisition of a heart's image over a cardiac cycle by storing 2DE images on video tape and then reconstructing and volume rendering them. 'Repetitive low-pass filtering' was used during reconstruction to fill the spaces between radial slices, which suppressed aliasing artifacts. [Tomographic Technologies 1991] provides flexible re-slicing by up to 6 planes as well other imaging modes. [Collet Billon 1990] uses two visualization techniques: re-slicing by an arbitrary plane and volume rendering. The former allows faster but only 2D viewing on a current workstation. The latter allows 3D viewing but often involves cumbersome manual segmentation. The reconstruction algorithm uses straightforward low pass filtering.

3. Incremental Volume Visualization

We have been experimenting with volume rendering as one alternative for visualizing dynamic ultrasound volume data. Standard volume rendering techniques which rely heavily on preprocessing do not apply well to dynamic data which must be visualized in real time [Levoy 1988; Sabella 1988; Upson 1988]. We review here an incremental , interactive, 3D ultrasound visualization technique which visualizes a 3D volume as it is incrementally updated by a sequence of registered 2D ultrasound images [Ohbuchi 1990; 1991].

Our target function is sampled at irregular points and may change over time. Instead of directly visualizing samples from this target, we reconstruct a regular 3D volume from this time series of spatially irregular sample points. This places a limit on storage and computation requirements which would grow without bound if we retained all the past sample points. The reconstructed volume is then rendered with an incremental volume-rendering technique.

The reconstruction is a 4D convolution process. A 3D Gaussian kernel is used for spatial reconstruction followed by a temporal reconstruction based on simple auto regressive moving average (ARMA) filtering [Haddad 1991]. Time stamps are assigned to each 3D voxel, which are updated during reconstruction. The time stamp difference between a reconstructed voxel and an incoming sample is used to compute coefficients for the ARMA filter. The 3D Gaussian filter is loosely matched to the point spread function of the ultrasound transducer and is a good choice because it minimizes the product of spatial bandwidth and spatial frequency bandwidth [Hildreth 1983; Leipnik 1960].

An image-order, ray-casting algorithm based on [Levoy 1988] renders the final images incrementally. Rendering is incremental and fast only if the viewpoint is fixed and if the updated volume is relatively small. Shading and ray sampling are done only for voxels proximate to incoming data. The ray samples are stored

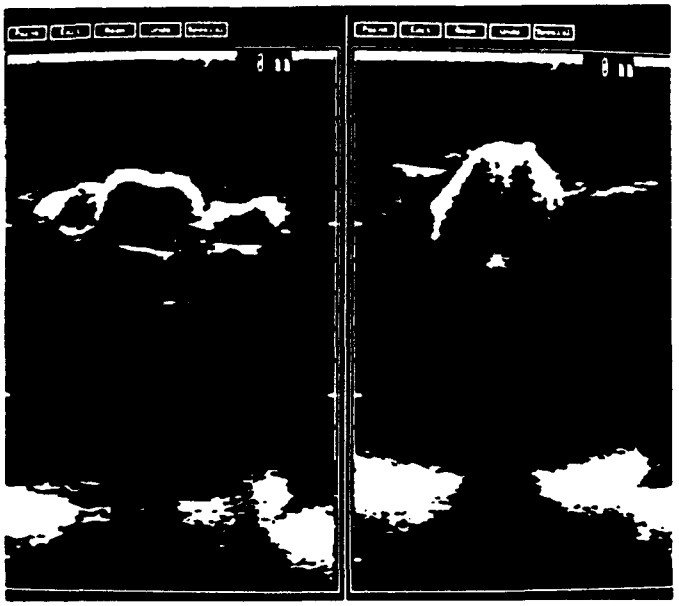


Figure 1. Two of 90 2D ultrasound echography images of a plastic toy doll phantom which was scanned in a water tank. The scans shown are at the torso (left) and at the head (right). The clouds at the bottom of the scans are artifacts due to reflections from the bottom of the water tank.

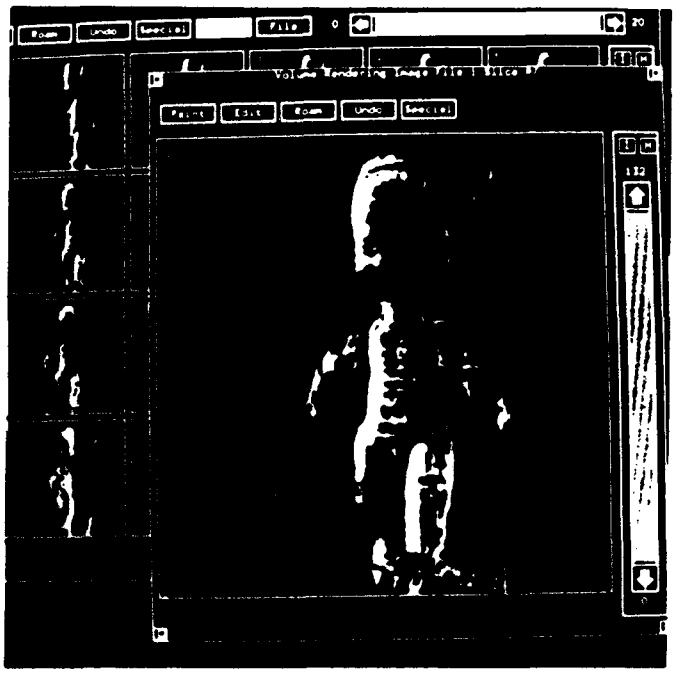


Figure 2. Reconstructed and rendered image of the toy doll phantom using incremental volume visualization.

in a 3D array in screen space called a "ray cache" for later use. The ray cache is hierarchical so that a small partial update of the ray cache can be composited quickly (O($log(n)$)) [Ohbuchi 1991]. The hierarchical ray cache also allows fast rendering of polygons properly composited with volume data, which can enhance the volume visualization [Levoy 1990; Miyazawa 1991]. This incremental volume rendering algorithm is not restricted to ultrasound and is applicable to other problems which update volume data incrementally, e.g., interactive volume modeling by sculpting [Galyean 1991].

To test this visualization technique, we acquired a series of 2D images with a manually guided conventional 2DE scanhead attached to a mechanical tracking arm with 3 DOF (two translations and one rotation). As we scanned various targets in a water tank, their images and their corresponding geometry were stored off-line. We then ran the incremental volume visualization algorithm on a DECstation 5000 with 256 MB of memory using this data. With a reconstruction buffer size of 150 × 150 × 300 and an image size of 256 × 256, it took 15–20 seconds to reconstruct and render a typical image after insertion of a 2D data slice. This time varied with reconstruction, shading, and viewing parameters.

Figure 1 shows 2 out of 90 2D images of a plastic toy doll phantom which is visualized in Figure 2. The 2D images were produced by an ATL Mark-4 Scanner with a 3.5 MHz linear scanhead. The 2D images overlap but are roughly parallel at approximately 2 mm intervals.

4. Virtual Environment Ultrasound Imaging

Various medical ultrasound imaging applications require a registration of ultrasound images with anatomical references, e.g., in performing a fine needle aspiration biopsy of a suspected breast tumor [Fornage 1990]. A virtual environment which displays images acquired by ultrasound equipment in place within a patient's anatomy could facilitate such an application. We have developed an experimental system that displays multiple 2D medical ultrasound images overlaid on real-world images. In January 1992, after months of development with test objects in water tanks, we performed our first experiment with a human subject.

Our virtual environment ultrasound imaging system works as follows (note that this is a different system than our older one described in the previous section): as each echography image is acquired by an ultrasound scanner, its position and orientation in 3D world space are tracked with 6 degrees of freedom (DOF). Simultaneously the position and orientation of a HMD are also tracked with 6 DOF. Using this geometry, an image-generation system generates 3D renderings of the 2D ultrasound images. These images are video mixed with real-world images from a miniature TV camera mounted on the HMD. The resulting composite image shows the 2D ultrasound data registered in its true 3D location.

Figure 3 is a block diagram of our system's hardware. There are three major components: 1) an image-acquisition and tracking system, which consists of an ultrasound scanner and a Polhemus tracking system, 2) an image-generation system, which is our Pixel-Planes 5 graphics multicomputer, and 3) a HMD which includes a portable TV camera, a video mixer, and a VPL EyePhone. Each component is described in more detail in Sections 4.1–4.3.

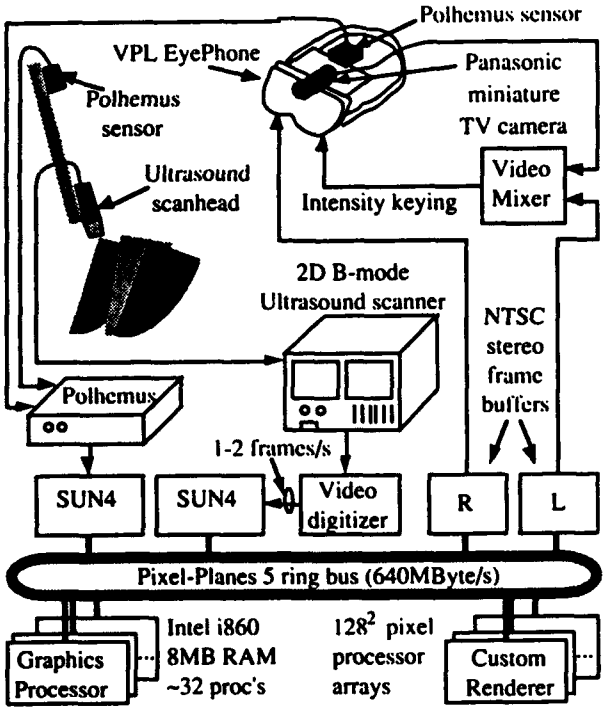


Figure 3. Hardware block diagram for the virtual environment ultrasound system.

4.1 Image Acquisition and Tracking

Two dimensional ultrasound images are generated by an IREX System III echography scanner with a 16 mm aperture 2.5 MHz phased array transducer. These images are digitized by a SUN 4 with a Matrox MVP/S real-time video digitizer and transferred to our Pixel-Planes 5 graphics multicomputer [Fuchs 1989]. The SUN 4 operates as a 2DE image server for requests from the Pixel-Planes 5 system. Images are distributed among the Graphics Processors (GPs) on a round-robin scan-line by scan-line basis. Due to the bandwidth limitations of the SUN 4 VME bus, transfer of the $512 \times 480 \times 8$ bits/pixel images is limited to 2 Hz.

A Polhemus system with one source and two receivers is used for tracking [Polhemus 1980]. One receiver tracks the HMD. The other tracks the ultrasound transducer. The Polhemus system is mounted in non ferrous materials away from magnetic interference sources such as the ultrasound transducer, HMD, and other lab equipment. A calibration procedure is used to relate both the ultrasound transducer to its Polhemus receiver and the HMD TV camera to its Polhemus receiver mounted on the HMD. This calibration procedure is described in Section 4.4.

4.2 Image Generation

Images are generated by the Pixel-Planes 5 system based on geometry information from the tracking system. Pixel-Planes 5 runs a custom PHIGS implementation which incorporates a facility to update display structures asynchronously from the display process. This separates the interactive virtual environment update rate from the 2D ultrasound image data acquisition rate. Images in the virtual environment are registered to the real world within the update-rate limit of the tracking and display system and not within the acquisition-rate limit of the image-acquisition system.

Pixels from the 2D ultrasound images are rendered as small, unshaded sphere primitives in the virtual environment. The 2D ultrasound images appear as space-filling slices registered in their correct 3D position. The ultrasound images are distributed among the GPs where they are clipped to remove unnecessary margins and transformed into sphere primitives, which are then sent to the Renderer boards for direct rasterization. Pixel-Planes 5 renders spheres very rapidly, even faster than it renders triangles, over 2 million per second [Fuchs 1985; 1989]. Final images are assembled in double buffered NTSC frame buffers for display on the HMD. To reduce the number of sphere primitives displayed, the ultrasound images are filtered and subsampled at every 4th pixel. Due to the low resolution of the HMD and inherent bandwidth limitation of the ultrasound scanner, this subsampling does not result in a substantial loss of image quality. An option to threshold lower intensity pixels in 2D ultrasound images prior to 3D rendering can suppress lower intensity pixels from being displayed.

4.3 Video See-Through HMD

A video see-through HMD system combines real-world images captured by head-mounted TV cameras with synthetic images generated to correspond with the real-world images. The important issues are tracking the real-world cameras accurately and generating the correct synthetic images to model the views of the cameras. Correct stereo modeling adds concerns about matching a pair of cameras to each other as well as tracking and modeling them. [Robinett 1991] discusses stereo HMD in detail and includes an analysis of the VPL EyePhone.

A Panasonic GP-KS102 camera provides monocular see-through capability for the left eye in our current system. Images from this camera are mixed with synthetic images from the Pixel-Planes 5 system using the luminance (brightness) keying feature on a Grass Valley Group Model 100 video mixer. With luminance keying, the pixels in the output image are selected from either the real-world image or the synthetic image, depending on the luminance of pixels in the synthetic image. The combined image for the left eye and a synthetic image only for the right eye are displayed on a VPL EyePhone.

4.4 Calibration

Two transformations, a "transducer transformation" and a "camera transformation," are needed to calibrate our test system. The transducer transformation relates the position and orientation of the Polhemus tracker attached to the ultrasound transducer to the position and scale of 2D ultrasound image pixels in 3D space. The camera transformation relates the position and orientation of the head-mounted Polhemus tracker to the HMD TV camera position, orientation, and field of view.

Both transformations are calculated by first locating a calibration jig in both the lab (real) and tracker (virtual) 3D coordinate systems. This is accomplished by performing rigid body rotations with the transducer tracker about axes which are to be fixed in both the real and virtual coordinate systems. Two samples from the tracker, each consisting of both a position and an orientation, are

sufficient to fix each calibration axis. The transducer transformation is computed by taking an ultrasound image of a target of known geometry placed at a known position on the calibration jig. By finding the pixel coordinates of point targets in the ultrasound image, the world coordinates of pixels in the ultrasound image can be found. From this relationship and the location of the Polhemus tracker attached to the ultrasound transducer at the time the target was imaged, the transducer transformation is derived. Similarly, the camera transformation is found by placing the HMD TV camera at known positions and orientations relative to the calibration jig. The field of view of the TV camera is known from camera specifications. Manual adjustments are used to improve the camera transformation.

5. Experimental Results

In January 1992 we conducted an experiment with a live human subject using the method described above. We scanned the abdomen of a volunteer who was 38 weeks pregnant. An ultrasound technician from the Department of Obstetrics & Gynecology of the UNC Hospitals performed the ultrasound scanning.

Figure 4 is a scene from the experiment. A person looks on with modified VPL EyePhone with the miniature video camera mounted on top and in front. Figure 5 shows the left eye view from the HMD, a composition of synthetic and real images. Figure 6 is another view from the left eye of the HMD wearer which shows several 2D ultrasound images in place within the subject's abdomen.

Figure 5. A video image presented to the left eye of the HMD showing a view of the subject's abdomen with a 2D ultrasound image superimposed and registered. Note the ultrasound transducer registered with the image acquired by it. The 2D image is from the antero-inferior view.

Figure 4. An ultrasound technician scans a subject while another person looks on with the video see-through head-mounted display (HMD). Note the miniature video camera attached to the front of the VPL EyePhone HMD.

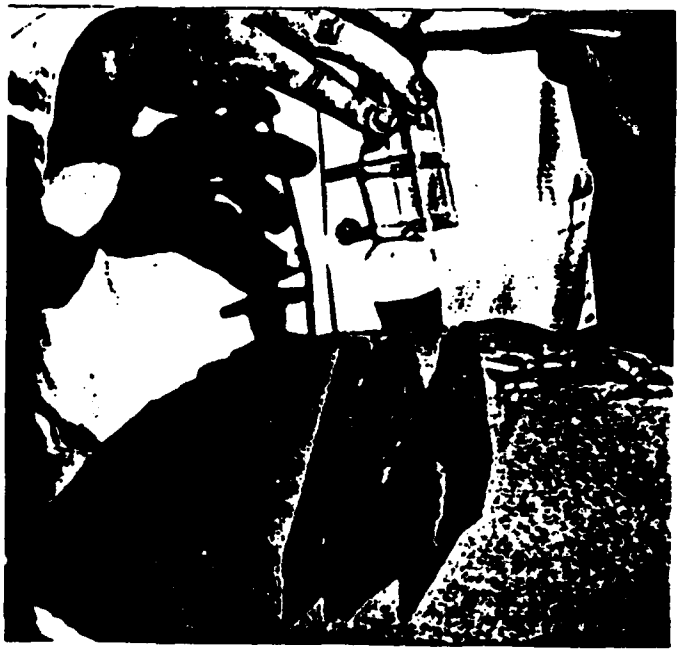


Figure 6. Another video image presented to the HMD showing several 2D image slices in 3D space within the patient's abdomen. The image slices are from the anterior view.

6. Conclusions and Future Directions

The results presented so far are the initial steps in the first application of what we hope will be a flourishing area of computer graphics and visualization.

6.1 Remaining Technical Problems

1) Conflicting visual cues: Our experiment (Figures 5 and 6) showed that simply overlaying synthetic images on real ones is not sufficient. To the user, the ultrasound images did not appear to be *inside* the subject, so much as pasted on *top* of her. To overcome this problem, we now provide additional cues to the user by making a virtual hole in the subject (Figure 7) by digitizing points on the abdomenal surface and constructing a shaded polygonal pit. The pit provides occlusion cues by obscuring the abdomenal surface along the inside walls of the pit. Shading the pit provides an additional cue. Unfortunately, this does not completely solve the problem; the pit hides *everything* in the real image that is in the same location (in 2D) as the pit, including real objects that are closer in 3D than the pit. (Note in Figure 7, the edge of the transducer is hidden behind the pit representation even though it should appear in front of it.)

To solve this problem, the systems needs to know depth information for both the real and synthetic objects visible from the HMD user's viewpoint. This would make it possible to present correct occlusion cues by combining the live and synthetic images with a Z-buffer like algorithm. An ideal implementation of this would require real-time range finding from the viewpoint of the HMD user - a significant technical challenge. Graphics architectures that provide real-time depth-based image composition are already under development [Molnar 1992].

Figure 7. An image showing a synthetic hole rendered around ultrasound images in an attempt to avoid conflicting visual cues. Note the depth cues provided by occlusion of the image slices by the pit walls and shading of the pit. Also note the incorrect obscuration of the ultrasound transducer by the pit wall. (RT3200 Advantage II ultrasound scanner courtesy of General Electric Medical Systems.)

Another remaining problem is the visualization of internal 3D structure in data captured by the ultrasound scanner. Neither our incremental volume rendering algorithm (Section 3) nor multiple explicit image slices in 3-space (Figure 6) solve this problem well. A combination of multiple visualization methods will probably be necessary in the future. We suspect that this problem is difficult because the human visual system is not accustomed to seeing structure within opaque objects, and so our development cannot be guided by the "gold standard" of reality that has been used so effectively in guiding other 3D rendering investigations.

2) System lag: Lag in image generation and tracking is noticeable in all head-mounted displays; but it is dramatically accentuated with see-through HMD. The "live video" of the observer's surroundings moves appropriately during any head movement but the synthetic image overlay lags behind. This is currently one of our system's major problems which prevents it from giving the user a convincing experience of seeing synthetic objects or images hanging in 3-space. A possible solution may be to delay the live video images so that their delay matches that of the synthetic images. This will align the real and synthetic images, but won't eliminate the lag itself. We are also considering predictive tracking as a way to reduce the effect of the lag [Liang 1991]. Developers of some multi-million dollar flight simulators have studied predictive trackingfor many years, but unfortunately for us, they have not, to our knowledge, published details of their methods and their methods' effectiveness. For the immediate future, we are planning to move to our locally-developed "ceiling tracker" [Ward 1992] and use predictive tracking.

3) Tracking system range and stability: Even though we are using the most popular and probably most effective commercially available tracking system from Polhemus, we are constantly plagued by limitations in tracking volume and tracking stability [Liang 1991]. The observer often steps inadvertently out of tracker range, and even while keeping very still the observer must cope with objects in the synthetic image "swimming" in place. We are eagerly awaiting the next generation of tracking systems from Polhemus and other manufacturers that are said to overcomemost of these problems. Even more capable tracking systems will be needed in order to satisfy the many applications in which the observer must move about in the real world instead of a laboratory, operating room or other controlled environment. Many schemes have been casually proposed over the years, but we know of no device that has been built and demonstrated. Even the room-size tracker we built and demonstrated for a week at SIGGRAPH'91 still needs special ceiling panels with infrared LEDs [Ward 1992].

4) Head-mounted display system resolution: For many of the applications envisioned, the image quality of current head-mounted video displays is totally inadequate. In a see-through application, a user is even more sensitive to the limitations of his head-mounted display than in a conventional non-see-through application because he is painfully aware of the visual details he's missing.

5) More powerful display engines: Even with all the above problems solved, the synthetic images we would like to see, for example, real-time volume visualization of real-time volume data, would still take too long to be created. Much more powerful image

generation systems are needed if we are to be able to visualize usefully detailed 3D imagery.

6.2 Other Applications

1) Vision in surgery: In neurosurgery, ultrasound is already used to image nearby arteries that should be avoided by an impending surgical incision.

2) Burning buildings: With close-range, millimeter wavelength radar, rescuers may be able to "see through" the smoke in the interior of burning buildings.

3) Building geometry: Geometry or other structural data could be added to a "live" scene. In the above "burning building" scenario, parts of a building plan could be superimposed onto the visual scene, such as the location of stairways, hallways, or the best exits out of the building.

4) Service information: Information could be displayed to a service technician working on complicated machinery such as a jet engine. Even simpler head-mounted displays, ones without head tracking, already provide information to users on site and avoid using a large cumbersome video screens. Adding head tracking would allow 3D superimposition to show, for instance, the location of special parts within an engine, or the easiest path for removal or insertion of a subassembly.

5) Architecture on site: Portable systems could allow builders and architects to preview buildings on site before construction or visualize additions to existing architecture.

With the work presented here and the identification of problems and possibilities for further research, we hope to encourage applications not only of "virtual environments" (imaginary worlds), but also applications that involve an "enhancement of vision" in our real world.

Acknowledgments

We would like to thank the following people: David Chen and Andrew Brandt for experimental assistance; General Electric Medical Systems (and especially R. Scott Ray) for the loan of an ultrasound scanner; Stefan Gottschalk for much assistance with video acquisition, editing, and printing; Professor Olaf von Ramm (Duke University) for donation of the IREX ultrasound scanner; ultrasound technician George Blanchard, RDMS, for scanning the subject; David Harrison and John Hughes for video and laboratory setup; Andrei State for experimental assistance; John Thomas for fabrication of a custom camera mount; Terry Yoo for video tape editing; Vern Katz, MD, for assistance with multiple ultrasound machines and scanning experiments; Nancy Chescheir, MD, for loan of an ultrasound machine and arrangements with the ultrasound technician; Warren Newton, MD, and Melanie Mintzer, MD, for finding our subject; Warren Robinett and Rich Holloway, for consultation with HMD optics and software; Professor Stephen Pizer and Charlie Kurak for consultation on the difficulty of enhancing ultrasound images; David Adam (Duke University) for instruction in the use of the IREX scanner; and our subject and her husband for their time and patience.

This research is partially supported by DARPA ISTO contract DAEA 18-90-C-0044, NSF ERC grant CDR-86-22201, DARPA ISTO contract 7510, NSF grant MIP-9000894, NSF cooperative agreement ASC-8920219, and NIH MIP grant PO 1 CA 47982, and by Digital Equipment Corporation.

References

[Brinkley 1978] Brinkley, J. F., Moritz, W.E., and Baker, D.W. "Ultrasonic Three-Dimensional Imaging and Volume From a Series of Arbitrary Sector Scans." *Ultrasound in Med. & Biol.*, 4, pp317-327.

[Collet Billon 1990] Collet Billon, A., *Philips Paris Research Lab.* Personal Communication.

[Fornage 1990] Fornage, B. D., Sneige, N., Faroux, M.J., and Andry, E. "Sonographic appearance and ultrasound guided fine-needle aspiration biopsy of brest carcinomas smaller than 1 cm^3." *Journal of Ultrasound in Medicine*, 9, pp559-568.

[Fuchs 1985] Fuchs, H., GoldFeather, J., Hultiquist, J.P., Spach, S., Austin, J., Brooks, Jr., F.P., Eyles, J., and Poulton, J. "Fast Spheres, Textures, Transparencies, and Image Enhancements in Pixel Planes." *Computer Graphics (Proceedings of SIGGRAPH'85)*, 19(3), pp111-120.

[Fuchs 1989] Fuchs, H., Poulton, J., Eyles, J., Greer, T., Goldfeather, J., Ellsworth, D., Molnar, S., and Israel, L. "Pixel Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories." *Computer Graphics (Proceedings of SIGGRAPH'89)*, 23(3), pp79-88.

[Galyean 1991] Galyean, T. A., and Hughes, J.F. "Sculpting: An Interactive Volumetric Modeling Technique." *Computer Graphics (Proceedings of SIGGRAPH'89)*, 25(4), pp267-274.

[Geiser 1982a] Geiser, E. A., Ariet, M., Conetta, D.A., Lupkiewicz, S.M., Christie, L.G., and Conti, C.R. "Dynamic three-dimensional echocardiographic reconstruction of the intact human left ventricle: Technique and initial observations in patients." *American Heart Journal*, 103(6), pp1056-1065.

[Geiser 1982b] Geiser, E. A., Christie, L.G., Conetta, D.A., Conti, C.R., and Gossman, G.S. "Mechanical Arm for Spatial Registration of Two-Dimensional Echographic Sections." *Cathet. Cariovasc. Diagn.*, 8, pp89-101.

[Ghosh 1982] Ghosh, A., Nanda, C.N., and Maurer, G. "Three-Dimensional Reconstruction of Echo-Cardiographics Images Using The Rotation Method." *Ultrasound in Med. & Biol.*, 8(6), pp655-661.

[Haddad 1991] Haddad, R. A., and Parsons, T.W. *Digital Signal Processing, Theory, Applications, and Hardware*. New York, Computer Science Press.

[Harris 1990] Harris, R. A., Follett, D.H., Halliwell, M, and Wells, P.N.T. "Ultimate limits in ultrasonic imaging resolution." *Ultrasound in Medicine and Biology*, 17(6), pp547-558.

[Hildreth 1983] Hildreth, E. C. "The Detection of Intensity Changes by Computer and Biological Vision Systems." *Computer Vision, Graphics, and Image Processing*, 22, pp1-27.

[Hottier 1989] Hottier, F., *Philips Paris Research Lab.* Personal Communication.

[Hottier 1990] Hottier, F., Collet Billon, A. *3D Echography: Status and Perspective*. 3D Imaging in Medicine. Springer-Verlag. pp21-41.

[Itoh 1979] Itoh, M., and Yokoi, H. "A computer-aided three-dimensional display system for ultrasonic diagnosis of a breast tumor." *Ultrasonics*, , pp261-268.

[King 1990] King, D. L., King Jr., D.L., and Shao, M.Y. "Three-Dimensional Spatial Registration and Interactive Display of

Position and Orientation of Real-Time Ultrasound Images." *Journal of Ultrasound Med*, 9, pp525-532.

[Lalouche 1989] Lalouche, R. C., Bickmore, D., Tessler, F., Mankovich, H.K., and Kangaraloo, H. "Three-dimensional reconstruction of ultrasound images." *SPIE'89, Medical Imaging*, pp59-66.

[Leipnik 1960] Leipnik, R. "The extended entropy uncertainty principle." *Info. Control*, 3, pp18-25.

[Levoy 1988] Levoy, M. "Display of Surface from Volume Data." *IEEE CG&A*, 8(5), pp29-37.

[Levoy 1990] Levoy, M. "A Hybrid Ray Tracer for Rendering Polygon and Volume Data." *IEEE CG&A*, 10(2), pp33-40.

[Liang 1991] Liang, J., Shaw, C., and Green, M. "On Temporal-Spatial Realism in the Virtual Reality Environment." *User Interface Software and Technology, 1991*, Hilton Head, SC., U.S.A., pp19-25.

[Lin 1991] Lin, W., Pizer, S.M., and Johnson, V.E. "Surface Estimation in Ultrasound Images." *Information Processing in Medical Imaging 1991*, Wye, U.K., Springer-Verlag, Heidelberg, pp285-299.

[Matsumoto 1981] Matsumoto, M., Inoue, M., Tamura, S., Tanaka, K., and Abe, H. "Three-Dimensional Echocardiography for Spatial Visualization and Volume Calculation of Cardiac Structures." *J. Clin. Ultrasound*, 9, pp157-165.

[McCann 1988] McCann, H. A., Sharp, J.S., Kinter, T.M., McEwan, C.N., Barillot, C., and Greenleaf, J.F. "Multidimensional Ultrasonic Imaging for Cardiology." *Proc.IEEE*, 76(9), pp1063-1073.

[Mills 1990] Mills, P. H., and Fuchs, H. "3D Ultrasound Display Using Optical Tracking." *First Conference on Visualization for Biomedical Computing*, Atlanta, GA, IEEE, pp490-497.

[Miyazawa 1991] Miyazawa, T. "A high-speed integrated rendering for interpreting multiple variable 3D data." *SPIE*, 1459(5),

[Molnar 1992] Molnar, S., Eyles, J., and Poulton, J. "PixelFlow: High-Speed Rendering Using Image Composition." *Computer Graphics (Proceedings of SIGGRAPH'92)*, ((In this issue)),

[Moritz 1983] Moritz, W.E., Pearlman, A.S., McCabe, D.H., Medema, D.K., Ainsworth, M.E., and Boles, M.S. "An Ultrasonic Techinique for Imaging the Ventricle in Three Dimensions and Calculating Its Volume." *IEEE Trans. Biom. Eng.*, BME-30(8), pp482-492.

[Nakamura 1984] Nakamura, S. "Three-Dimensional Digital Display of Ultrasonograms." *IEEE CG&A*, 4(5), pp36-45.

[Ohbuchi 1990] Ohbuchi, R., and Fuchs, H. "Incremental 3D Ultrasound Imaging from a 2D Scanner." *First Conference on Visualization in Biomedical Computing*, Atlanta, GA, IEEE, pp360-367.

[Ohbuchi 1991] Ohbuchi, R., and Fuchs, H. "Incremental Volume Rendering Algorithm for Interactive 3D Ultrasound Imaging." *Information Processing in Medical Imaging 1991 (Lecture Notes in Computer Science, Springer-Verlag)*, Wye, UK, Springer-Verlag, pp486-500.

[Pini 1990] Pini, R., Monnini, E., Masotti, L., Novins, K. L., Greenberg, D. P., Greppi, B., Cerofolini, M., and Devereux, R. B. "Echocardiographic Three-Dimensional Visualization of the Heart." *3D Imaging in Medicine*, Travemünde, Germany, F 60, Springer-Verlag, pp263-274.

[Polhemus 1980] Polhemus. *3Space Isotrak User's Manual*.

[Raichelen 1986] Raichelen, J. S., Trivedi, S.S., Herman, G.T., Sutton, M.G., and Reichek, N. "Dynamic Three Dimensional Reconstruction of the Left Ventricle From Two-Dimensional Echocardiograms." *Journal. Amer. Coll. of Cardiology*, 8(2), pp364-370.

[Robinett 1991] Robinett, W., and Rolland, J.P. "A Computational Model for the Stereoscopic Optics of a Head-Mounted Display." *Presence*, 1(1), pp45-62.

[Sabella 1988] Sabella, P. "A Rendering Algorithm for Visualizing 3D Scalar Fields." *Computer Graphics (Proceedings of SIGGRAPH'88)*, 22(4), pp51-58.

[Shattuck 1984] Shattuck, D. P., Weishenker, M.D., Smith, S.W., and von Ramm, O.T. "Explososcan: A Parallel Processing Technique for High Speed Ultrasound Imaging with Linear Phased Arrays." *JASA*, 75(4), pp1273-1282.

[Smith 1991] Smith, S. W., Pavy, Jr., S.G., and von Ramm, O.T. "High-Speed Ultrasound Volumetric Imaging System - Part I: Transducer Design and Beam Steering." *IEEE Transaction on Ultrasonics, Ferroelectrics, and Frequency Control*, 38(2), pp100-108.

[Stickels 1984] Stickels, K. R., and Wann, L.S. "An Analysis of Three-Dimensional Reconstructive Echocardiography." *Ultrasound in Med. & Biol.*, 10(5), pp575-580.

[Thijssen 1990] Thijssen, J. M., and Oosterveld, B.J. "Texture in Tissue Echograms, Speckle or Information ?" *Journal of Ultrasound in Medicine*, 9, pp215-229.

[Tomographic Technologies 1991] Tomographic Technologies, G. *Echo-CT*.

[Upson 1988] Upson, C., and Keeler, M. "VBUFFER: Visible Volume Rendering." *ACM Computer Graphics (Proceedings of SIGGRAPH'88)*, 22(4), pp59-64.

[von Ramm 1991] von Ramm, O.T., Smith, S.W., and Pavy, Jr., H.G. "High-Speed Ultrasound Volumetric Imaging System - Part II: Parallel Processing and Image Display." *IEEE Transaction on Ultrasonics, Ferroelectrics, and Frequency Control*, 38(2), pp109-115.

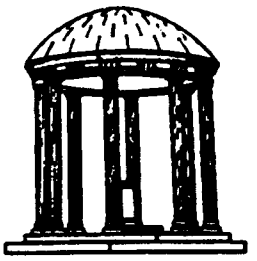
[Ward 1992] Ward, M., Azuma, R., Bennett, R., Gottschalk, S., and Fuchs, H. "A Demonstrated Optical Tracker with Scalable Work Area for Head-Mounted Display Systems." *1992 Symposium on Interactive 3D Graphics*, Cambridge, MA., ACM, pp43-52.

[Wells 1977] Wells, P. N. T. *Biomedical ultrasonics*. London, Academic Press.

Viper: A Quasi-Real-Time Virtual-Worlds Application

TR92-004
December 1991

Richard L. Holloway



The University of North Carolina
Chapel Hill, NC 27599-3175

The research was supported by the following grants: DARPA #DAEA 18-90-C-0044, NSF Cooperative Agreement #ASC-8920219, and DARPA: "Science and Technology Center for Computer Graphics and Scientific Visualization", ONR #N00014-86-K-0680, and NIH #5-R24-RR-02170.

Viper: A Quasi-Real-Time Virtual-Worlds Application

Richard L. Holloway

December 3, 1991

Abstract

Virtual-worlds applications have a great need for interactive performance, since the user's view of the real world may be blocked out and the only "world" the user can see is the virtual world created by the application. For this reason, the system's response is critical- if the user turns his head and the world takes seconds to catch up with him, the result will be disorienting and confusing, rather than enlightening. The *viper* application is an attempt at giving real-time or near-real-time performance in a virtual-worlds application; it allows the user to trade off image quality for interactive performance.

1. Overview

The overall goal of this project was to implement an adaptive system for approximating real-time performance with a graphics system that is overloaded. Since the underlying operating system is a non-real-time version of Unix, there was no way for an application-level system to guarantee real-time performance. Thus, the best solution was to create a system that monitors its own performance and adapts its task load according to how much it was able to get done previously.

Such a system was implemented, and the results were good: The *viper* system delivers interactive performance at the expense of image quality, with a user-controllable method for trading back performance for image quality.

2. The Problem

The general problem is that interactive graphics applications tend to overload the image-generation system, which means that the system's interactive performance suffers. There is typically a relationship between image complexity/quality and the time required to render the image. In general, better-looking images take more time. For an interactive application, this load may vary depending on how many polygons (or other graphics primitives) happen to fall within the field of view of the user. There are times when image quality is of the utmost importance, and so the user is prepared to wait for a high-quality image. Other times, the user just needs a coarse image in order to guide his interaction, so response time is the critical factor.

3. Relevant Previous Work

One solution to the conflict between image quality and interaction proposed by [Bergman et al. 86] is called *adaptive refinement*. The basic premise is that the user is guiding the navigation with an input device such as a joystick and viewing it on an ordinary CRT. Given this sort of system, it is possible to implement the system such that while the user is moving the joystick, simple images will be drawn in real time; when the user stops moving the joystick, successively better versions of the image will be drawn until either the user moves the joystick again, or the best-quality image has been generated.

While this technique is useful for *improving* interactive performance, it does nothing to *guarantee* performance. There is no notion of real time in this system, only a sequence of steps to be performed until they are finished or interrupted. The minimal image consisted of 10% of the image's vertices displayed at whatever update rate the graphics engine could muster (which for the 8,029-vertex test case they used was about 1 Hz on a Masscomp workstation).

Also, one of the key assumptions of this work is that there is some way to tell when the user has paused to see a particular view. With mice and joysticks, one can set a "movement delta"- a minimum amount that the joystick value has to change before it will be detected as true movement rather than noise on the A/D channel. With the head-mounted display, even small head movements are significant, since the user needs an up-to-date view of the world in order to keep himself oriented. Thus, the idea of incorporating a movement delta does not seem promising for this application. A compromise solution is the "slow motion" button, which is discussed below.

4. The Project

4.1 Overview

The basic idea is to use image database management and a notion of real time to achieve a specified update rate. Since there is no low-level operating-system support for a true real-time system, this system uses an adaptive scheme to approximate real-time performance. The application monitors its performance on the previous frame and uses knowledge about that frame to decide how much to attempt for the current frame. Since the system is non-preemptive, there is clearly a risk of failure- the system load could change in unpredictable ways so that the current frame's decision was always always wrong. Without any constraints on the external environment, there is no way to avoid this possibility; the best we can do is to hope that we can predict it well enough to get good performance.

As a compromise solution to the adaptive refinement approach, I implemented a "slow motion button" under user control: When the user presses this button, full-quality images are generated at whatever (slower) rate this requires. Releasing the button speeds things up again.

4.2 The Application

The application for testing this approach is a head-mounted display (HMD) system running a modified version of the *vixen* program. *Vixen* is a model-viewing HMD application that allows a user with a HMD to walk or fly around any object defined in the PPHIGS archive format. The user can also grab the object and scale it. An informal sketch of *vixen* follows.

```
main()
{

init;
read_archive_file;

while ( True )
    {
    read_tracker;
    update_head_and_hand;
    check_for_events;
    draw_frame;
    }
}
```

At present, *vixen* reads in the archive file and displays a complete image at every frame, no matter how long it takes. The archive file is made up of graphics primitives such as polygons, spheres, colors and transformations, and is organized into a *display list,* which is traversed at each call to *draw_frame.*

4.3 The New System

In order to get better performance, a new system was created to a) segment the models and b) display them in real time. Originally I proposed to create a single program to read in an archive file, divide it into independent segments, and display them. I ended up dividing these two tasks into two standalone programs, since one often wants to make many runs with the same data set and does not want the overhead of segmenting the data set every time. The segmenting program was called *seg,* and the display program was called *viper.*

One of the things that shaped the course of the implementation was the nature of the models selected as typical for this application. A model with many unique features (such as a space station) requires much more careful pruning than one made up of many homogeneous polygons (such as a model of human lungs). Luckily, my driving research problem involves medical data sets in which almost all of the polygons are small and equally (un)important. For this reason, I ended up choosing a very simple priority scheme and a very simple adaptation scheme. I did run the system on a space station model as well, and it worked well enough, but not quite as well as with the medical models.

4.3.1 *Seg*

There were two main versions of *seg*- the first version followed the original plan of dividing the image up into separate, independent segments. To do this, I wrote a lexical analyzer and a yacc parser for reading in PPHIGS archive files in ASCII format. PPHIGS has entities called *structures* which are collections of graphics primitives, color, matrix and other commands. A graphical object is typically represented as one or more structures.

Seg's job was to look through the structures in the archive file and find large groups of polygons (called *spans*) that it could divide up into new segments. The segments were put into new structures in the output stream.

Once the object was divided into many structures, it was easy for the display program to add and remove them from the display list at run-time according to the load on the system. The problem with this approach was that the overhead for a structure is significant, so that converting a file with *seg* sometimes created so much overhead just for the structures that the system became overloaded and the performance was poor. Using fewer structures per original structure helped, but the Pixel-Planes group suggested a better alternative, so this approach was scrapped.

The second version used a different idea: instead of dividing the original, large structures into a bunch of smaller ones, use "variable-size" structures. PPHIGS on Pxpl5 has a structure primitive called *conditional_return* which allows the display control code to stop processing a structure based on a global system flag. An example follows.

```
structure example {

    polygon { <coords> };
    polygon { <coords> };
    polygon { <coords> };

    cond_return flag_value1

    polygon { <coords> };
    polygon { <coords> };
    polygon { <coords> };

    cond_return flag_value2

    polygon { <coords> };
    polygon { <coords> };
    polygon { <coords> };
};
```

When the *example* structure is traversed by the display code, *flag_value* is logically ORed with a user-settable system flag, and the return is executed if the result is non-zero. Since there are 32 bits in this system flag, each structure can have 32 conditional returns inserted into it, allowing the display code to break out of it at any of these 32 points. This strategy allows the display code to go as deep into each structure as time permits while keeping the overhead low.

The second version of *seg* was similar to the first, except that instead of dividing large structures into many smaller ones, it output the original number of structures with conditional returns inserted. The conditional returns effectively divided the structures into segments. In addition, *seg* randomized the polygons in the spans so that all parts of the object were represented in each of the segments.

4.3.2 *Viper:* Vixen with Interactive PERformance

The second part of the system was a modified version of *vixen* that accepted a desired frame-rate argument and did timing calculations and load-balancing to achieve the desired frame rate.

As proposed, I made the following modifications to *vixen:*

- Added a command line argument for specifying the desired frame rate. If none was specified, *viper* used a value of 20 frames per second.
- Added code for calculating the current frame rate and adjusting the display list accordingly.
- Added a "slow motion" button so that the user could see all of the primitives in the image at a slower frame rate.
- Added code to keep track of the system's performance over time. At the end of session, the system prints out the following information:
  - Frame rate data: desired frame rate, mean actual frame rate, standard deviation about the mean, low and high values. Also, the percentage of frames at or above the desired frame rate and the percentage of frames at or above 75% of the desired frame rate.
  - Segment data: Total number of segments, average number of segments displayed, standard deviation, and low value.
  - Frame rate and number of segments for every frame printed to a file in format suitable for viewing with *xgraph.*

Some parts of the original proposal were "thrown overboard" during the implementation process. Strangely enough, this was not due to running out of time (as it usually is) as much as the fact that these ideas just didn't fit in with the final implementation. All of the following were in the "time-permitting" category anyway.

Differences from original proposal:

- The priority scheme was very simple. There were two priorities: High and low. High priority objects (such as the hand and various environment objects) were always drawn; low priority objects were drawn in order as time permitted. This scheme, although simple, was completely adequate for the application, so other priority schemes were not investigated (nor were they really feasible).
- Costs were not calculated for segments. This is related to the priority discussion above, but basically, since the model was made up of many small polygons and the segments were roughly the same size, there was no need to try to calculate costs.

- Ordered vs. random add/delete operations were not implemented because of the linear nature of the conditional returns; ie., if you use conditional returns, the only choice your program has is when to exit the structure- you have no way of skipping the early parts and executing only the latter parts. The ordered add/delete strategy works very well anyway, so this was not a major limitation.

- Predictive tracking was not done. As the graphs in the experimental section show, the system adapted quickly to changes in the load, so there was no real need for the added complexity of predictive tracking.

- Display estimation errors were not displayed as a graph in screen space; instead, the data was output to a file and viewed with *xgraph*. This was not only much easier to implement, it made more sense since the VPL EyePhone's screens have such horrible resolution (worse than 200x200).

5. Why *Viper* is a Real-Time System

Strictly speaking, it's not. Since this system is built on top of Unix, the operating system may go out to lunch at any time and all *viper's* deadlines will be left in the dust. I would call this system *pseudo-real-time*, since it emulates a real-time system. *Viper* has a notion of guaranteed performance rather than deadlines- it does not set up a timer and try to get things done before the time goes off; rather, it does a certain amount of work and then evaluates its *past* performance in order to decide what to attempt for the future.

In other ways, it *is* a real-time system because it puts responsiveness first- it is trying to meet deadlines even though it does not have complete control over whether it makes them. *Viper* is at the mercy of its external environment, but so are most useful real-time systems. Finally, it is a real-time system because, by and large, it gives real-time performance. As shown in the Experiments section, the frame rate delivered is at or better than the requested frame rate around 90% of the time.

6. Lessons Learned

- The overhead of trying to make an application perform better may actually make it perform worse (this is what happened with the first version of *seg* when I tried dividing large structures into hundreds of small structures).

- Models with objects made up of only a few large polygons must have higher priority- you really miss them when they blink out. This strategy is best suited for models with lots of small polygons. Priorities based on polygon size could alleviate this limitation.

- Even though the system's clock had a granularity of one microsecond, the frame rates were tended to have much higher granularity. Jumps in frame time were always around 10ms, probably due to Pixel-Planes 5's internal cycle time. This granularity caused a similar granularity in the image, since primitives are thrown out until a certain frame rate is reached.

- The system worked even though no adjustments were made on the host processing end- all of the slack was taken up by varying the load on the graphics subsystem.

- Trying to adjust the load as quickly as possible did not work. In other words, if the system suddenly became heavily loaded, dropping a bunch of polygons all at once instead of doing it gradually failed. It failed because for transient variations, the system would add or subtract a bunch of polygons very suddenly, sending it into oscillation and producing large, abrupt image changes. Since there were only 32 levels per structure, dropping or adding them one at a time was reasonable. Doing them two at a time did not noticeably improve performance.

7. Experiments

I did several sample runs with the system to evaluate its performance. I tried both static (no appreciable head motion, and thus no major image complexity changes) and dynamic (significant load changes) sessions, both with and without the use of the slow motion button. I also varied the load from light loads (a few hundred polygons) to heavy loads (tens of thousands of polygons).

General Explanations of Data

- The term "level" in the data refers to the number of segments displayed (PPHIGS refers to these as "refinement levels").

- The graphs display both level information (dotted line) and frame rate information (solid line). The horizontal axis gives the frame count. The vertical indicates both frame rate and level. Specific data on the session is given underneath the graph.

- The program initially starts out with all 32 levels displayed, then adjusts the number dynamically. When the "slow motion" button is pressed, the number of levels is set to 32 for as long as the button is held, which is shown by the dotted line going abruptly up to 32 (except under light loads, where it is already at or near 32) and staying flat.

- The frame rate is initially estimated to be approximately 20, so the frame rate curve always starts there.

- Within the normal system oscillations, there are occasional peaks and valleys caused by the Unix load which further exercise the system.

- The "light load" was about 550 polygons; the "heavy load" was about 18,000, except for the long interactive session, which had 37,000.

The actual data follows the conclusion section.

8. Conclusion

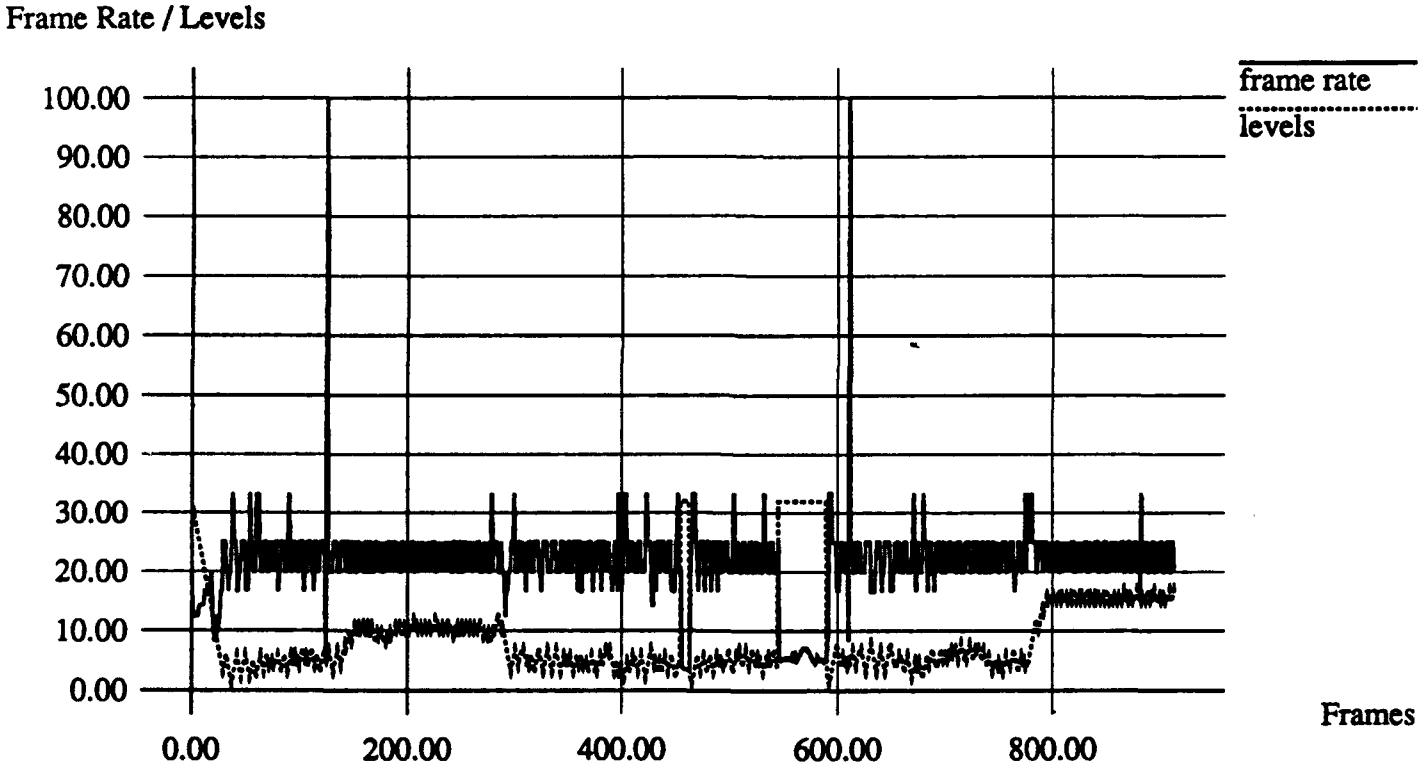
The system worked better than I had expected. I had feared that the overhead of telling the graphics system which segments to drop (and the segments themselves) would end up costing as much as it saved, or I would have to optimize both the Unix side and the graphics side in order to get good performance. The system has been a pleasant surprise and may well work its way into the next version of *vixen*, or even into the x-ray vision system for my dissertation.

9. Acknowledgements

This research was supported by the following grants: DARPA #DAEA 18-90-C-0044, NSF Cooperative Agreement #ASC-8920219, and DARPA: "Science and Technology Center for Computer Graphics and Scientific Visualization", ONR #N00014-86-K-0680, and NIH #5-R24-RR-02170.

Explanation: This longer session had the user actually walking around the model and using the slow motion button (once briefly at 450 frames, then again from 550 to 600 frames). The dark stripe above the 20Hz line shows that the frame rate was usually above the requested rate.

Long Interactive Session w/ 37,000 Polygons



Frame Rate Data:

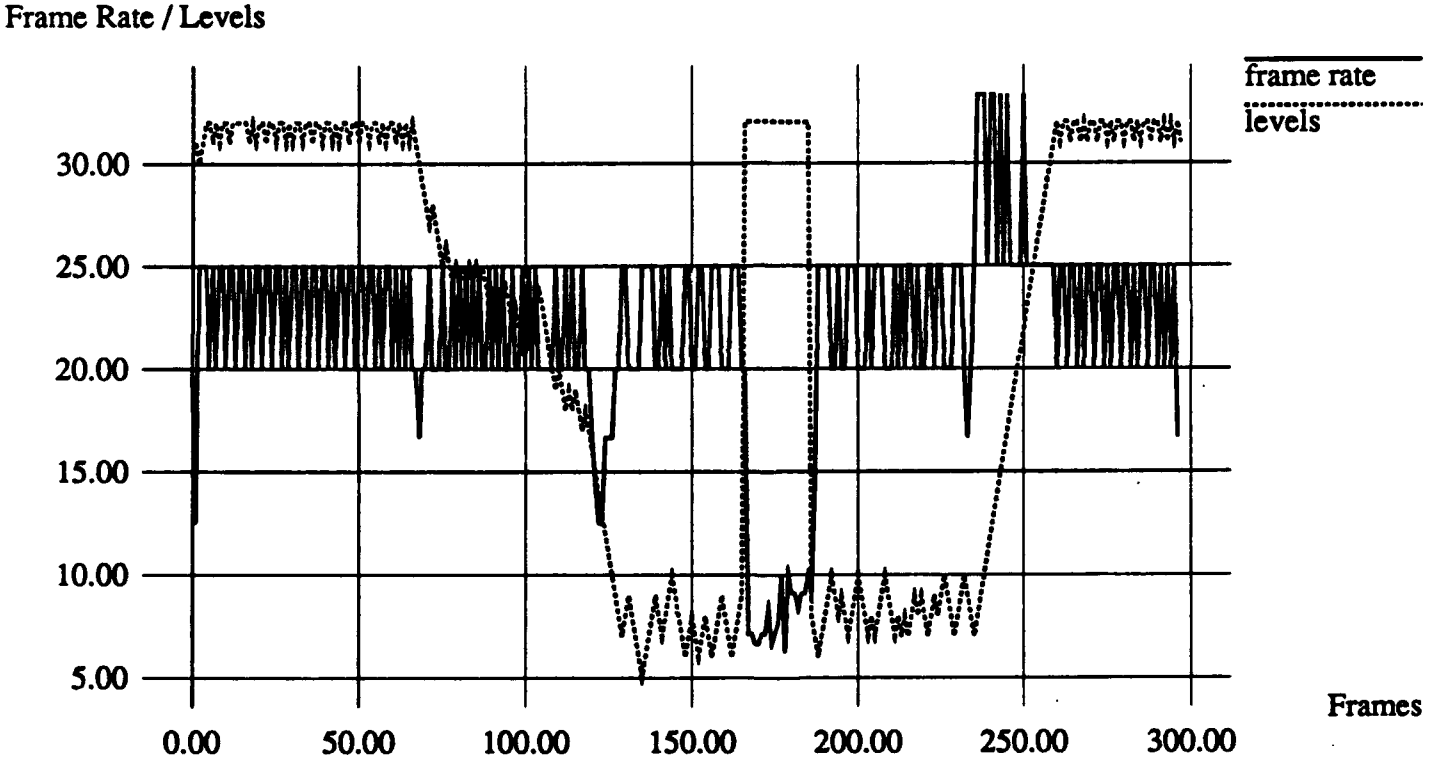
number of frames: 913; (approx. 45 seconds)
desired rate: 20.00
mean: 21.47 SD: 8.02
low: 3.45; high: 100.02
percentage at or above desired rate: 85.87%
percentage at or above 75% of desired rate: 91.57%

Level Data:

total possible: 32
mean: 9.21 SD: 7.86
low: 1

Explanation: In this interactive session, the user is initially looking away from the polygon cluster, then looks at it (as the level decreases) then presses the slow motion button, then releases it, then looks away. The load and frame rate vary accordingly.

Dynamic Heavy Load with Slow-Motion Button



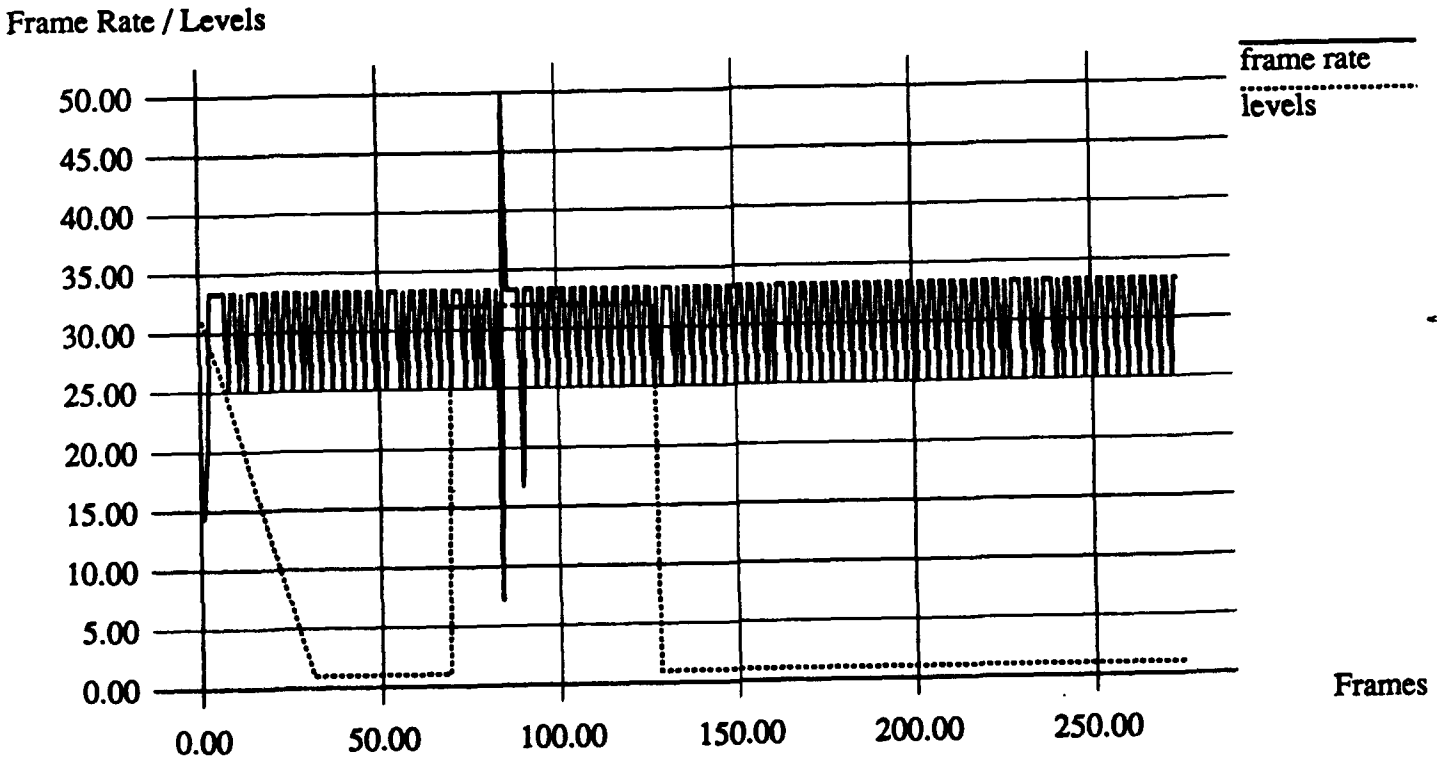
(look away, look at cluster, slow-mo, look at cluster, look away)

Frame Rate Data:
number of frames: 297; (approx. 14 seconds)
desired rate: 20.00
mean: 21.72 SD: 6.79
low: 6.25; high: 33.33
percentage at or above desired rate: 88.22%
percentage at or above 75% of desired rate: 91.58%

Level Data:
total possible: 32
mean: 21.60 SD: 11.35
low: 5

Explanation: The desired frame rate is set here to an impossibly high value (50). The system cannot meet this performance level and keeps dropping the level to try to improve performance. The level bottoms out at 1 (the minimum) except when the slow motion button is pressed.

Static Light Load, Frame Rate = 50, Slow-Motion Button

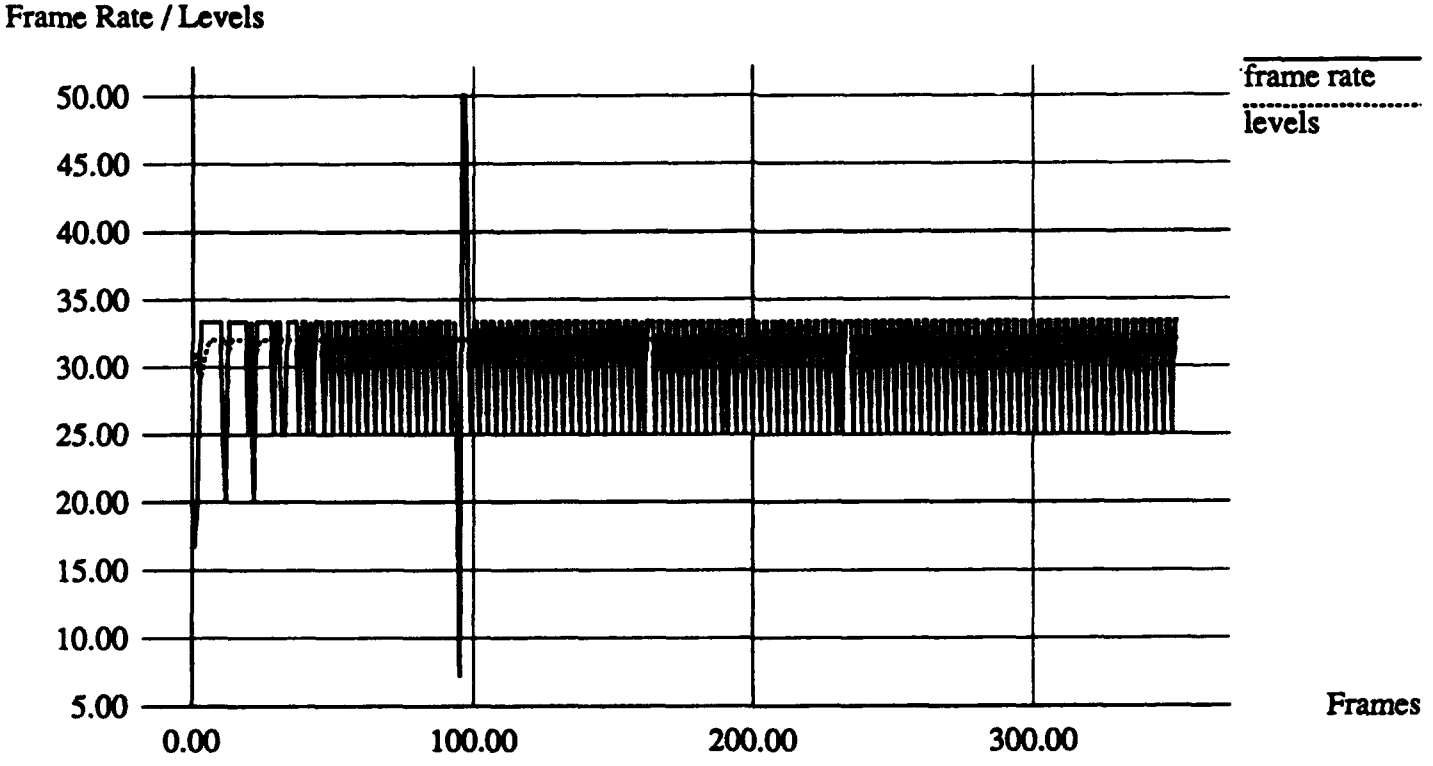


Frame Rate Data:
number of frames: 275; (approx. 5 seconds)
desired rate: 50.00
mean: 30.44 SD: 7.15
low: 7.14; high: 50.01
percentage at or above desired rate: 0.36%
percentage at or above 75% of desired rate: 0.36%

Level Data:
total possible: 32
mean: 9.23 SD: 13.36
low: 1

Explanation: With a light load, the slow motion button has practically no effect, since the level is already around 32.

Static Light Load, Slow-Motion Button



Frame Rate Data:

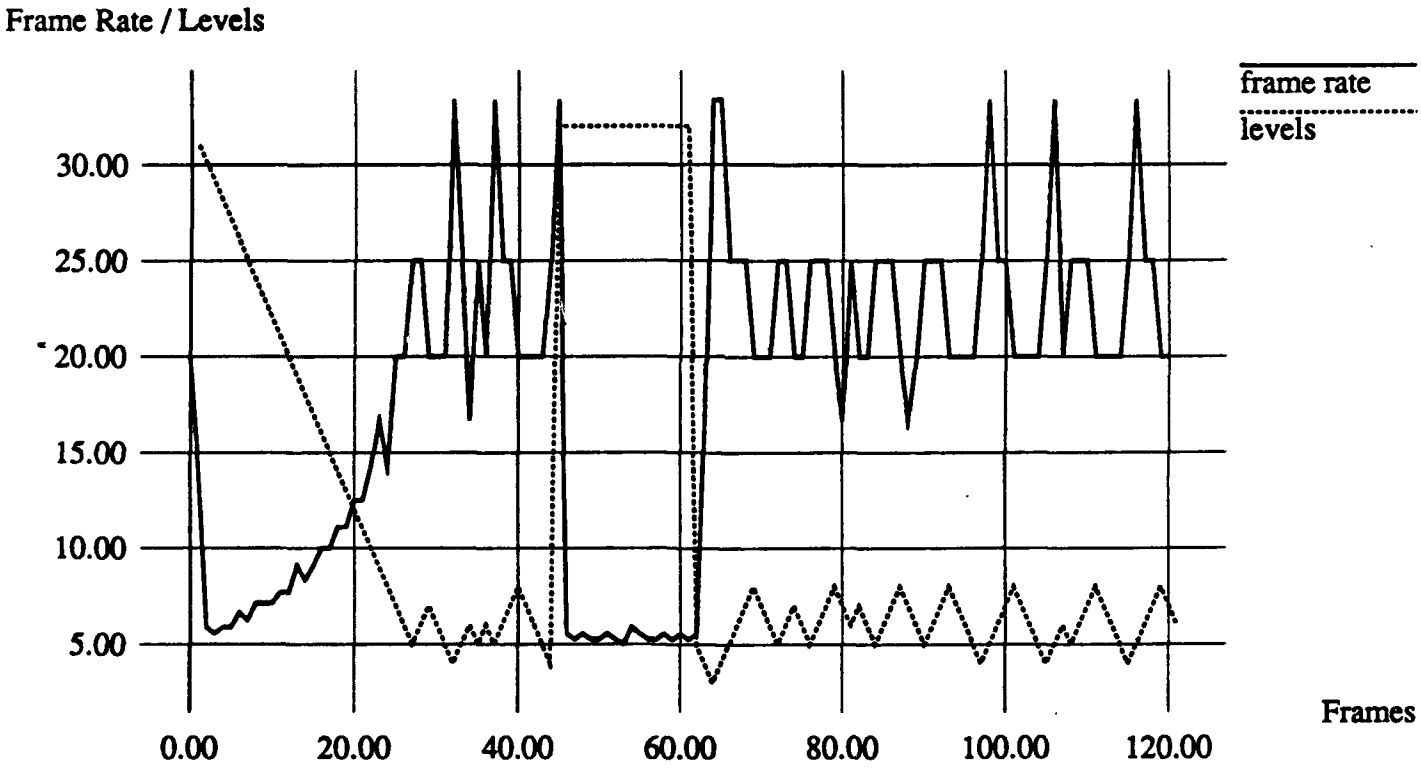
number of frames: 352; (approx. 17 seconds)
desired rate: 20.00
mean: 30.55 SD: 7.13
low: 7.14; high: 50.01
percentage at or above desired rate: 99.15%
percentage at or above 75% of desired rate: 99.72%

Level Data:

total possible: 32
mean: 31.97 SD: 5.66
low: 29

Explanation: Here, the HMD is always looking at a large cluster of polygons, but at about the 45th frame, the slow motion button is held down for about 17 frames, which sends the level up to 32 and the frame rate down to about 5. Once the button is released, the system returns to its previous state.

Static Heavy Load, Slow-Motion Button



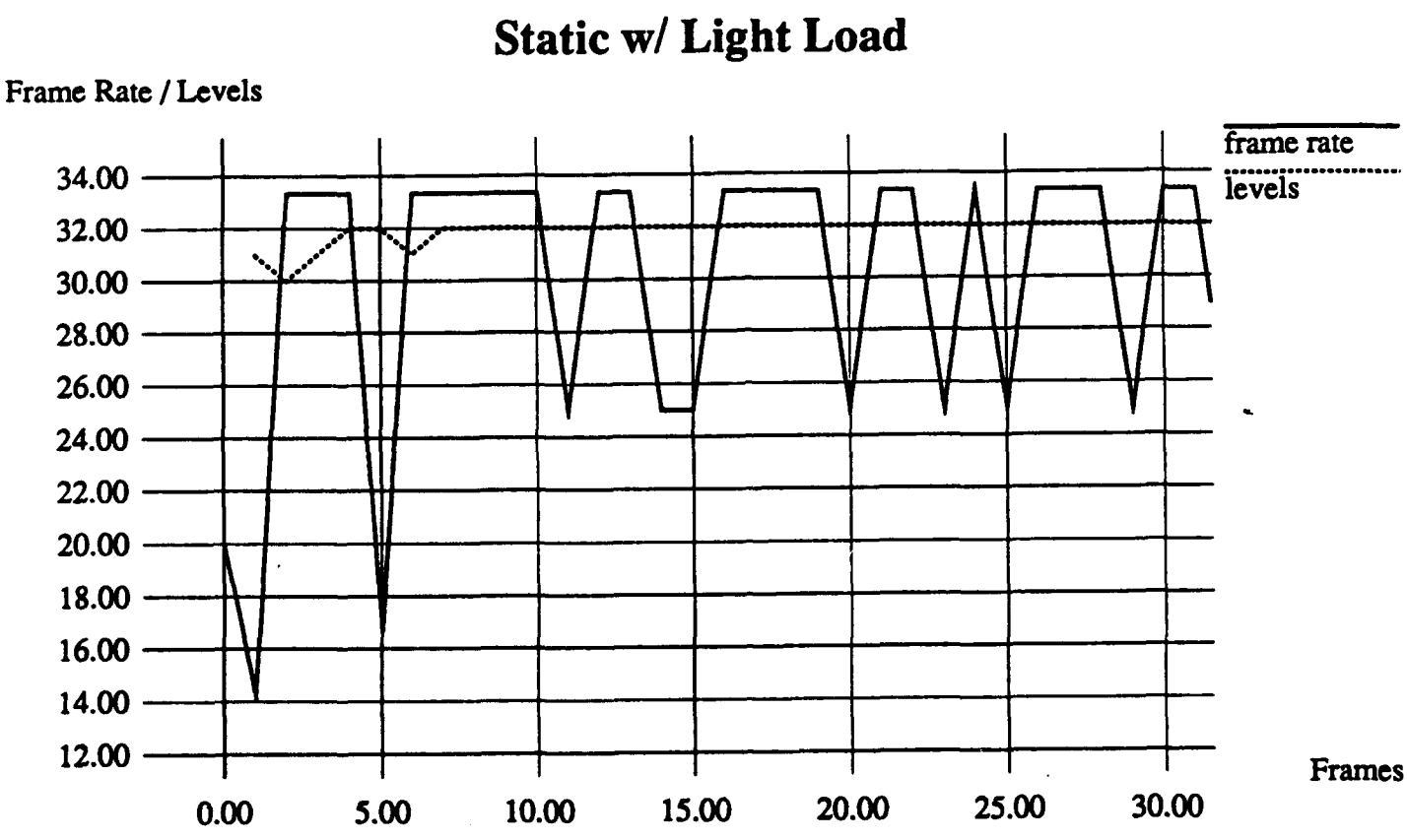
Frame Rate Data:

number of frames: 121; (approx. 6 seconds)
desired rate: 20.00
mean: 17.95 SD: 9.30
low: 5.00; high: 33.33
percentage at or above desired rate: 61.98%
percentage at or above 75% of desired rate: 66.94%

Level Data:

total possible: 32
mean: 12.33 SD: 10.67
low: 3

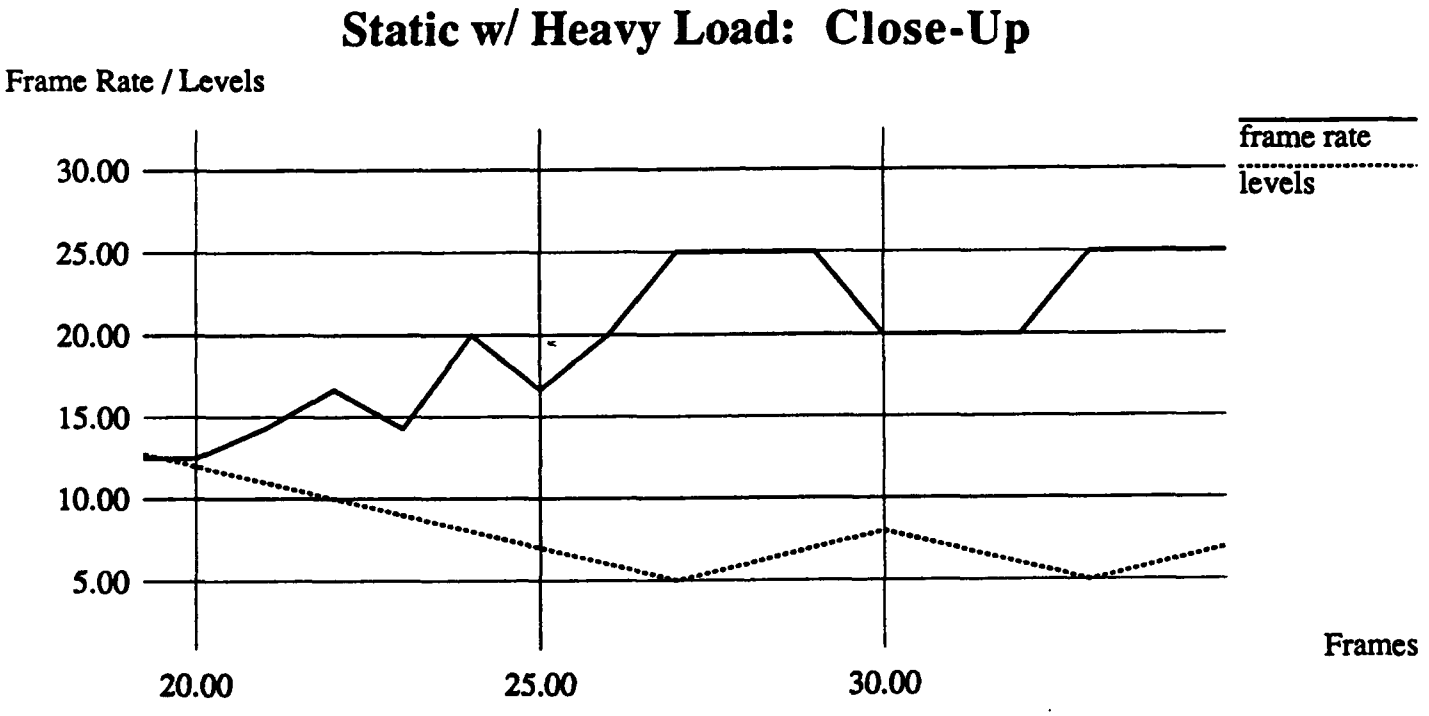
Explanation: With a light load, the system quickly achieves and holds the desired frame rate with the level at 32. Oscillation from 25Hz to 33.3Hz is probably due to varying system load.



Frame Rate Data:
number of frames: 220; (approx. 11 seconds)
desired rate: 20.00
mean: 30.42 SD: 6.93
low: 14.29; high: 33.33
percentage at or above desired rate: 99%
percentage at or above 75% of desired rate: 100%

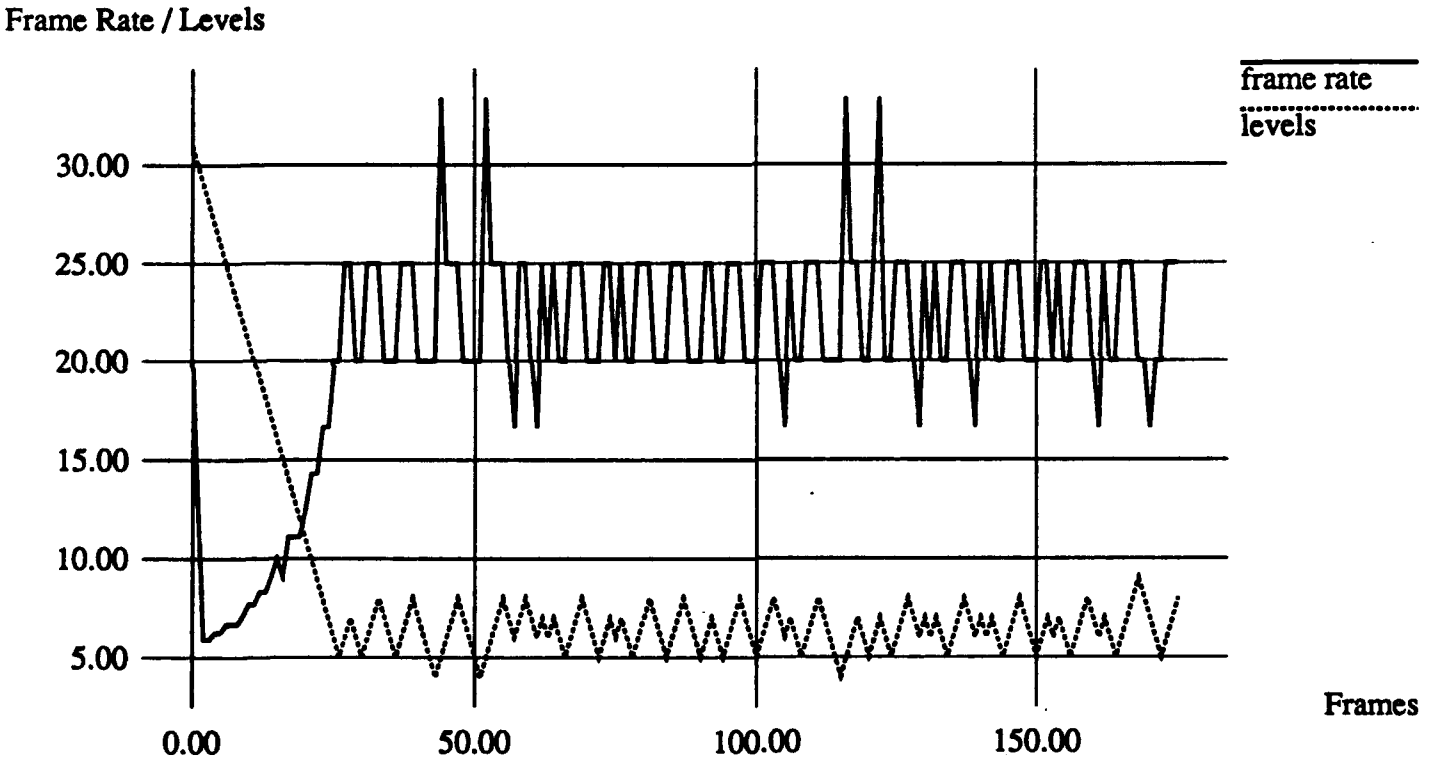
Level Data:
total possible: 32
mean: 31.98 SD: 5.66
low: 30

Explanation: Close-up of the previous graph at point where frame rate begins to clear the 20Hz threshold. Note that when the frame rate drops from 25 to 20, the system drops the level until it goes back up to 25, then increases the level until it drops back to 20. This oscillation guarantees that the system is always doing as much as it can.



Explanation: HMD was held still with large cluster of polygons in center of field of view. The level goes from 32 quickly to 5 or so, then oscillates between 4 and 8. The frame rate starts at 20 and quickly falls to 6, then climbs up to 20 and oscillates between 20 and 25.

Static w/ Heavy Load



Frame Rate Data:

number of frames: 176; (approx. 8 seconds)
desired rate: 20.00
mean: 20.77 SD: 7.12
low: 5.88; high: 33.34
percentage at or above desired rate: 79%
percentage at or above 75% of desired rate: 88%

Level Data:

total possible: 32
mean: 8.19 SD: 5.98
low: 4

A Short Guide to AutoCAD Drawing Primitives for 3D Computer Graphics Models and the Walkthrough AutoCAD-to-Polygon Conversion Program

John C. Alspaugh
22 April 1992

Abstract: Intended for those who want to build models for 3D computer graphics as painlessly as possible, this document introduces the reader to those AutoCAD drawing primitives and AutoCAD structures that can be converted to polygons with the Walkthrough AutoCAD-to-polygon conversion program. It also discusses those primitives that are not converted and why they are not converted. In addition, this document introduces the reader to more advanced AutoCAD modelling techniques for efficient model-building and model-management. The final section tells how to run the AutoCAD-to-polygon conversion program: file locations, input file requirements and forms, and the format of the output file.

Contents

AutoCAD: *n.* a large, complex computer-aided design program that strikes fear into the hearts of even the most experienced graphics users.

You've heard this before. Not the actual words, and probably not in dictionary entry style, but you have gotten the word that AutoCAD is the ultimate in ugly programs. This is somewhat exaggerated; it is a large program, with its share of strange commands and methods, but it is also an adequate tool for building 3-dimensional, architectural-type models.

This document is to help you get around these two difficulties. It details how to draw with AutoCAD, while ignoring the commands for manipulating the drawing, the drawing files, and the requisite hardware. I will discuss the major object manipulation and modification commands, but not the details of how to *run* AutoCAD. More complete treatments of AutoCAD are to be found in the *AutoCAD Reference Manual*, which sits on the lower shelf of the bookcase in the Graphics Lab between the aquarium and Cezanne. The *AutoCAD Reference Manual* is not very readable, and there are other manuals on the same shelf that are easier to follow. If you need more information, any of these documents can provide it.

Section 1, Entities, and Section 2, Blocks, discuss the creation, modification, and manipulation of entities and blocks, the AutoCAD drawing primitives. In Section 3, I will go into some advanced modeling techniques and, in Section 4, how to convert an AutoCAD DXF file (called a Drawing Exchange File and produced using the DXFOUT command in AutoCAD) into a file of polygons that you can further convert to whatever

format you need. We will do the conversion with an AutoCAD file parser called acadtopoly, built by the UNC Walkthrough project (for simplicity I will call the AutoCAD-to-polygon conversion program "the AutoCAD parser" or simply "the parser" from now on).

For greater clarity, AutoCAD commands will be in small caps, new terms and concepts will be in italics, Walkthrough program commands will be in a different font, and familiar terms applied in new context will be put in quotation marks. Boldface is used to give special emphasis where special emphasis is due.

1 Entities

AutoCAD has a number of primitives -- things such as lines, circles, and solids -- that make up a drawing. These primitives tend to be 2-dimensional entities -- lines, circles, polylines -- that can be extruded perpendicular to the plane; for example, a circle drawn in the X-Y plane, when extruded along the Z-axis, becomes a cylinder. The amount of extrusion is called the entity's *thickness*. Entities are simple objects that AutoCAD treats as single elements when creating, manipulating, or modifying them. Models are made by combining these primitives or collections of these entities called *blocks*. A block is a user-defined group of entities that can be treated as a single unit; I will discuss blocks at greater length in Section 2. An AutoCAD drawing is a list of blocks and primitives, each of which can be selected, manipulated, or modified without affecting the others. I will refer to these drawing primitives as *entities*, and will discuss their creation, manipulation, and modification in AutoCAD with attention to the way the parser interprets the AutoCAD commands.

I will discuss entity creation in greater detail than entity modification or entity manipulation,because the effects of entity modification are largely transparent to the parser and the effects of entity manipulation are entirely transparent. Modifying or manipulating an AutoCAD entity tends not to affect the parser's polygonal representation for that entity, just its position, orientation, color, thickness, and so on. In Sections 1.2 and 1.3, I will discuss what side effects do exist.

1.1 Entity creation

I will not dwell on how to draw the simple entities; the *AutoCAD Reference Manual*, chapter 4 (pages 65-110) covers this adequately. Entities are created with commands of the same name as the entity; for example, ARC creates an arc, SOLID a solid, CIRCLE a circle. The majority of these commands are self-documented and intuitive, with each step in the command indicating the sort of input that is expected. However, there are exceptions; I will detail how to draw some of the more unusual objects, because their methods are unexpected.

One thing to remember when drawing with AutoCAD: use the Right-hand Rule (Right-hand Rule: if the fingers of the right hand curl in the direction of the vertices -- the first vertex at the base of the finger, the last vertex at the finger tips -- then the thumb indicates the front of the face). Entities drawn by selecting points in a counterclockwise order will produce correctly oriented polygons. Figure 1 shows a back-facing polygon and a correctly oriented (or front-facing) polygon to illustrate this idea. The numbers beside the vertices give the order in which the vertex was drawn.

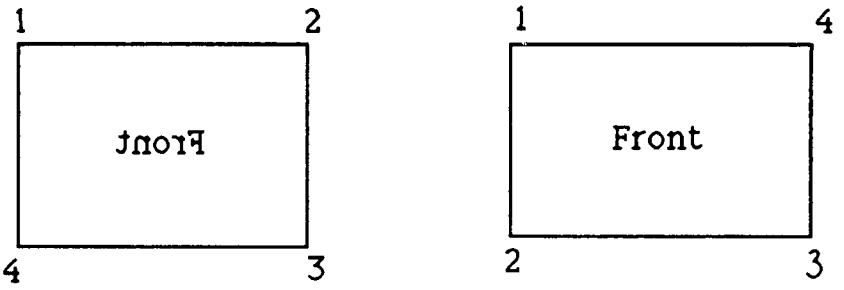


Figure 1: 3Dfaces. The face on the left is drawn with vertices chosen in clockwise order; the one on the right is drawn according to the Right-hand Rule.

There is, of course, an exception to this rule; the solid is drawn entirely differently (this will be discussed in Section 1.1.1.5).

1.1.1 Usable entities

The parser interprets some AutoCAD entities as you would expect it to, and others slightly differently. I will term *usable* entities those AutoCAD entities that the

parser correctly converts to polygons, with the polygons positioned as they were in the AutoCAD drawing. Those entities that the parser does not create polygons for, or does not make the polygons as they were in AutoCAD, will be termed *degenerate* entities and will be discussed throughout Sections 1.1.1 and 1.1.2.

1.1.1.1 ARC

The parser can produce two shapes from arcs; the resulting shape depends on whether the arc has a thickness (whether the arc is extruded perpendicular to the plane in which it is defined). A flat, 2-dimensional arc in the AutoCAD drawing becomes a flat, 2-dimensional surface: a surface enclosed by the curve and the chord across the end of it. It does not produce a pie shape, as one might expect. Figure 2 illustrates this with a conceptual drawing of two example arcs: an arc of less than 180 degrees, on the left, and an arc of more than 180 degrees, on the right. However, this is not how the actual polygons would be defined for such arcs. The parser would produce quadrilaterals **approximating** this shape, as in Figure 3, but not a single polygon **of** this shape (the number of quadrilaterals depends on the size of the angle).

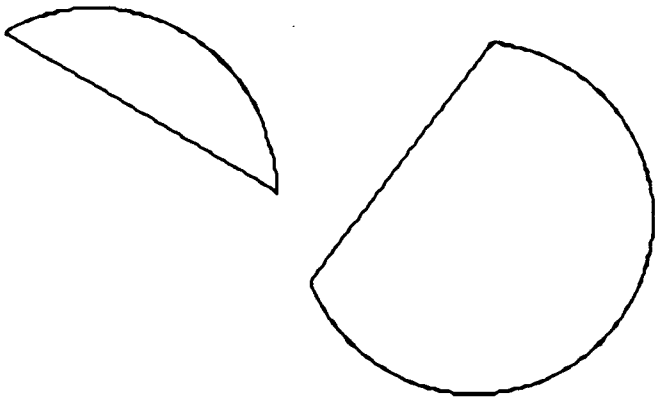


Figure 2: Conceptual Diagram of Faces Produced from Non-extruded Arcs

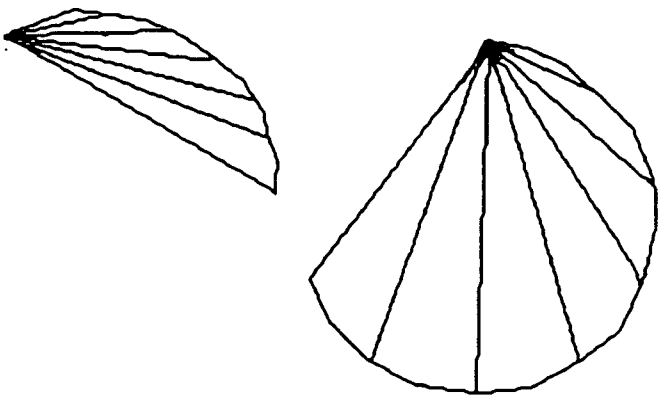


Figure 3: Quadrilaterals approximating the arcs of Figure 2

An extruded arc, one with thickness, produces polygons approximating a curved face around the axis of extrusion. It remains an arc, but is an arc with height. Figure 4 gives an example of how an extruded arc would appear after being converted by the parser, both conceptually and polygonally.

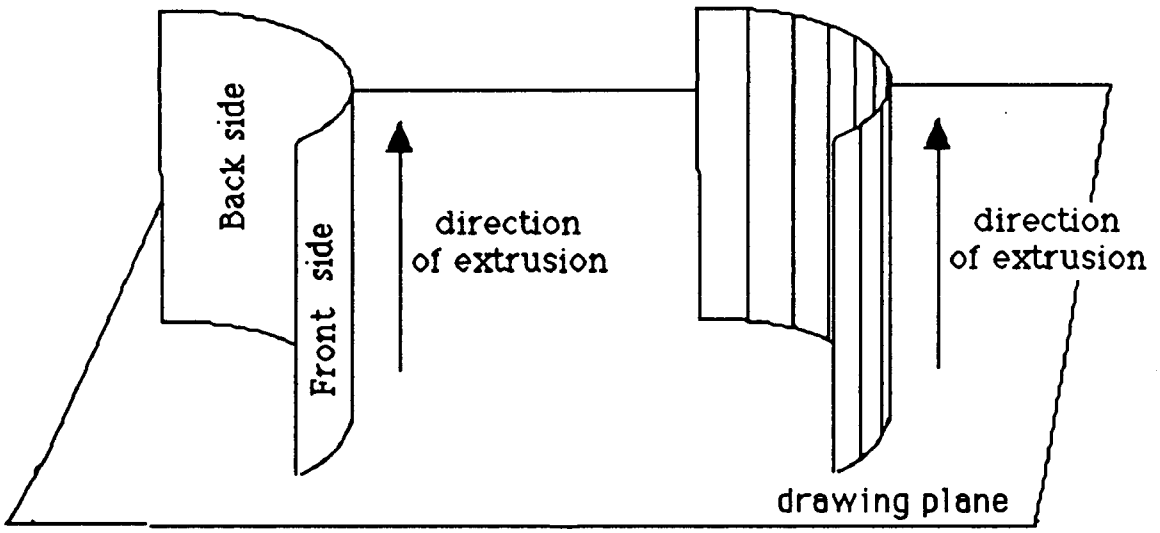


Figure 4: On the left, a conceptual diagram of an extruded arc, and on the right, the polygons produced by the parser. The view is at an oblique angle to the drawing plane.

The thickness can be set before creating the entity in AutoCAD (with the ELEV command) or, after the entity has been created, by modifying the entity with the CHANGE command (CHANGE THICKNESS). Entities that can be extruded (lines, arcs, polylines, and solids) are extruded perpendicularly to the plane in which they were

drawn. The 3Dfaces in Figure 1 and the arcs in Figure 2 (none of which are extruded) were drawn in the plane of this page. However, in Figure 4, the extruded arc, the drawing plane is skewed in relation to the plane of the page. If we were to view the extruded arc as it was drawn, we would see a curve with no discernable thickness. For clarity, views of extruded AutoCAD entities will be shown at an oblique angle, unless otherwise noted.

The sign of the thickness (negative or positive) influences the orientation of the polygons approximating the extruded arc. Figure 4, for example, has a positive thickness, as indicated by the direction of extrusion. Thus, the face in Figure 4 has its front side on the outside of the arc. A negative thickness would produce the reverse; the outside of the arc would be the back side of the face and the inside of the arc would be the front. This is illustrated in Figures 5 and 6. The line segment is the same in both figures; vertex 1 is the starting point of the segment, vertex 2 is the end point. The parser calculates the vertices of the polygon for this segment to be: vertex 1, vertex 2, vertex 2 + thickness, vertex 1 + thickness. A positive thickness means the polygon is defined as in Figure 5: the arrow inside the polygon indicates the order for the vertices, which is counterclockwise, and thus we are looking at the front of this polygon (by the Right-hand Rule). Likewise, in Figure 6, we are looking at the back side of the polygon because the vertices are listed in clockwise order. Extruded lines and polylines also share this property; the orientation for any segment of an extruded arc, line, or polyline is decided by the sign of the thickness.

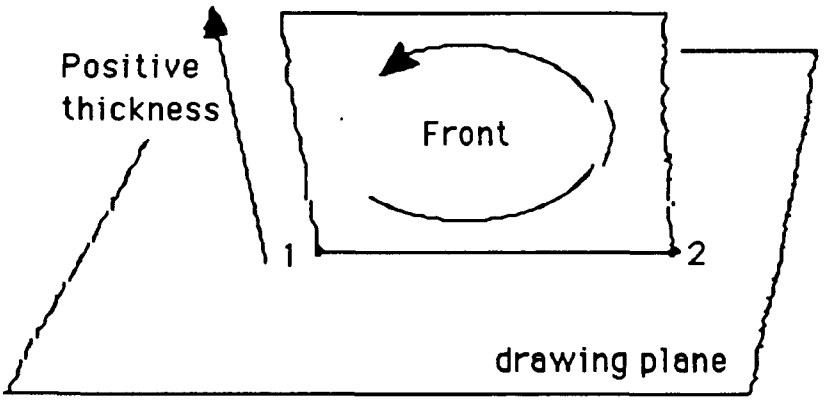


Figure 5: A Line Segment with Positive Thickness, viewed from an oblique angle. The axis of extrusion is coming out of the page.

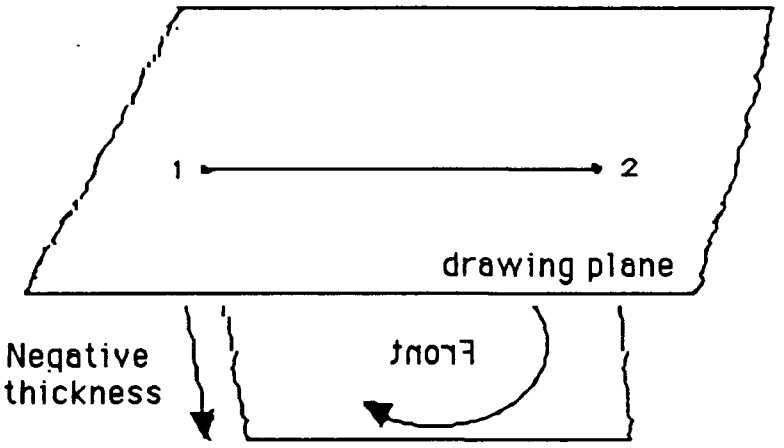


Figure 6: A Line Segment with Negative Thickness, viewed from an oblique angle. The axis of extrusion is going into the page.

1.1.1.2 Lines: LINE and PLINE

The LINE command creates useful entities only when given a thickness; a line without thickness is only a 2-dimensional object and cannot represent anything physical in 3-dimensional space. The parser thus ignores lines without thickness as degenerate entities. LINE with thickness produces a face in the direction of extrusion (lines are extruded perpendicularly to the the plane they were drawn in). Lines may be drawn as connected "segments," with the first vertex of the new segment beginning at the last vertex of the previous segment. These segments are not actually components of a single entity; each segment is itself an independent line. AutoCAD automatically gives the option of making the end point of the last segment the start point of a new segment. Each segment is manipulated or modified independently of the rest. This makes the line ideal for drawing walls in architectural models: they are quickly drawn, easily modified, and readily make wall-like polygons.

Polylines (produced with the PLINE command) are compound entities. A polyline is made of line segments and arcs in a plane, connected at end points. Although -- as in the line -- each line segment or each arc or the polyline is specified as beginning where the last ended, the polyline -- unlike the line -- is treated as a single entity: the segments are not independent. To select a segment of a polyline is to select the entire polyline. This cohesiveness makes them convenient for defining 3-dimensional meshes, in which one might want a compound curve to approximate a complex surface. As with the line,

the parser interprets only the 2-dimensional, extruded polyline as polygons, ignoring the 2-D polylines without thickness.

As in the extruded arc, the orientation of a line or polyline depends on the extrusion vector (whether the line has a positive or negative thickness). A positive thickness has an extrusion vector pointing above the plane ("up") the entity was defined in, and a negative thickness has the extrusion vector pointing below the plane ("down"). The parser interprets lines and the arcs and line segments of a polyline as depicted in Figures 4 and 5 of Section 1.1.1.1.

There is a 3-dimensional polyline called 3DPOLY, which can be defined in 3-dimensional space as opposed to a plane; however, the parser treats all cases of the 3DPOLY as degenerate and ignores them.

1.1.1.3 CIRCLE

The CIRCLE is similar to the arc; it can become either a flat face or an extruded shape. Without a thickness, it is a circular array of 12 triangles, as in Figure 7. Unlike previous figures, Figure 7 is not a conceptual diagram but a drawing of how the polygon approximation would look. The circular face is oriented with the front sides facing out of the screen as the circle was drawn. With positive thickness, the circle is interpreted as a hollow column, approximated by 12 polygons, with front faces facing outward, as in Figure 8. Figure 8 is also drawn as it is represented polygonally, rather than as a conceptual drawing.

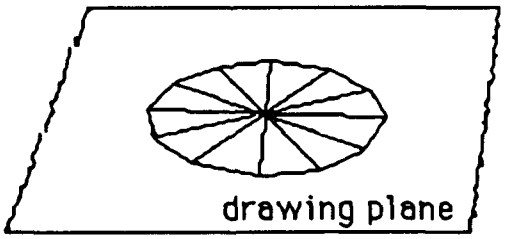


Figure 7: A Non-extruded Circle on a plane, viewed at an angle to demonstrate that it is without thickness.

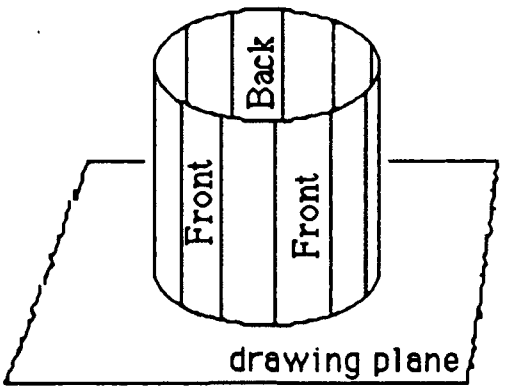


Figure 8: The Circle of Figure 7, extruded above the plane. A negative thickness would cause the circle to be extruded beneath the plane.

1.1.1.4 3DFACE

The 3dface, shown in Figure 1, is the AutoCAD entity most like the abstract idea of a polygon: it is a face, a polygon positioned in 3-dimensional space. It has, at most 4 vertices, and is selected a vertex at a time. It adheres to the Right-hand Rule and may be drawn in any 3-dimensional plane. This makes it ideal for sloping surfaces, such as roofs; it is also a good choice for drawing floor polygons. It cannot be extruded. 3Dfaces should be drawn according to the Right-hand Rule to ensure they are oriented correctly.

As far as the parser is concerned, 3Dfaces and extruded lines are frequently indistinguishable: both are converted into polygons. However, in the AutoCAD model, they are quite different. A line can only be extruded along the Z-axis, whereas a 3Dface can be oriented anywhere in 3-dimensional space. This implies that a polygon generated from an extruded line must be a rectilinear quadrilateral (two sets of parallel sides), because the extrusion is the same at both ends of the line. A 3dface is not constrained to have any parallel sides, or even four sides: you can, by hitting return instead of the fourth vertex, draw a triangle. But the 3dface is difficult to modify. The geometry can be changed only with the STRETCH command, which can be tedious in a complex model. It is much easier to change a line's thickness or start and end points (using CHANGE THICKNESS or TRIM and EXTEND , respectively). In the Walkthrough project models, lines are used for the walls and 3Dfaces for the window sills, roof, and furniture.

As a final note on lines versus 3Dfaces: it is possible to extrude lines other than along the Z-axis. There is a concept of "world" space, with a canonical set of axes, and within the world space, a user-defined set of axes where the user is drawing. This is the User Coordinate System (UCS), and it makes it possible to reorient the X-Y drawing plane in some arbitrary manner and then work there, as if in world space. UCS is beyond the scope of this primer; see p. 225 of the *AutoCAD Reference Manual* for instructions. UCS is not required for basic AutoCAD modeling, and can become very complicated to maintain. The parser does correctly position in world space entities drawn in a User Coordinate System.

1.1.1.5 SOLID

The solid is a quirky entity. Unlike most AutoCAD entities, which must be drawn counterclockwise to ensure proper orientation, solids are drawn in AutoCAD as in Figure 9. The numbers beside the vertices represent the order in which the vertices were drawn. Like the 3Dface, the solid has a maximum of 4 vertices (AutoCAD calculates the other vertices necessary to make a 3-D solid by adding the thickness to the Z value for each vertex. The X and Y for each vertex remains the same).

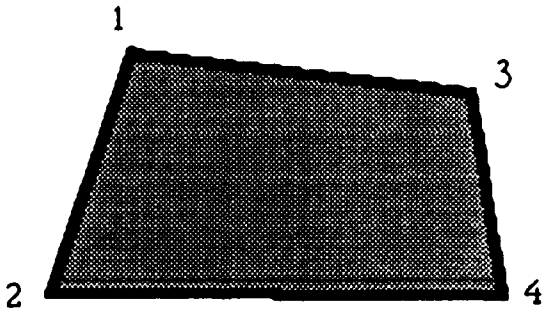


Figure 9: A Correctly Drawn Solid, as Seen from the Top. The drawing plane is parallel to this page.

A solid drawn according to the right-hand rule will produce a bow tie shape, as in Figure 10. This mistake cannot be corrected except by deleting the tie-shaped solid and redrawing the entity.

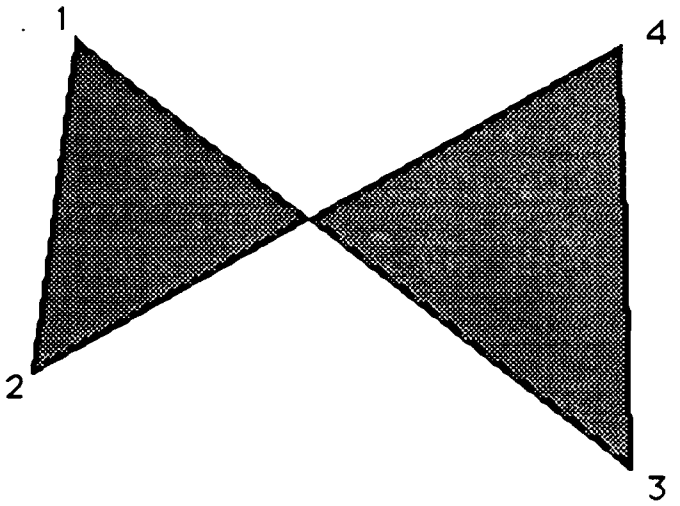


Figure 10: An Incorrectly Drawn Solid, Viewed from the Top. The drawing plane is parallel to this page.

The parser turns a solid into 6 polygons -- 4 sides, a top, and a bottom --that together define the same shape as in the solid, with the front sides on the exterior. Solids should be given a thickness; a solid without a thickness becomes two co-planar polygons, which is undesirable.

1.1.1.6 Meshes

Meshes are somewhat unusual entities, so I will go into more detail. There are four varieties of mesh-type entities, each doing something slightly different. The parser transforms the meshes into quadrilateral polygons almost exactly as one would expect, except when the mesh produces polygons that are not completely flat. These "bent" polygons are called **non-planar**, because they do not lie completely in any one plane. Non-planar polygons are undesirable (for example, the Walkthrough display programs will not load a non-planar polygon) and are automatically triangulated in the parser.

The four basic varieties of meshes are RULESURF, TABSURF, REVSURF, and EDGESURF. RULESURF, as in Figure 11, produces a mesh between two curves; TABSURF (Figure 12) makes a mesh from a curve defining a surface and an extrusion vector for that curve, extruding the curve the length of the vector in the direction the vector points; REVSURF (Figure 13) creates a surface of revolution by rotating a curve about an axis; EDGESURF (Figure 14) interpolates a mesh between four adjoining curves, which must meet at their end points. A curve in any of the surfaces can be an

ARC, LINE, PLINE, or 3DPOLY (3-dimensional polyline); these curves are separate from the mesh and, if not removed, will be interpreted by the parser as an arc, line, polyline, or 3D polyline, as is appropriate.

The number of gradations in the mesh are defined by the variables SURFTAB1 and SURFTAB2 (the default value for both variables is 6, and can be changed with the SETVAR <variable name> command). The parser translates meshes into polygons influenced by the SURFTAB variables; EDGESURF and REVSURF, for example, are converted into surfaces of SURFTAB1 times SURFTAB2 polygons. RULESURF and TABSURF are only SURFTAB1 number of polygons.

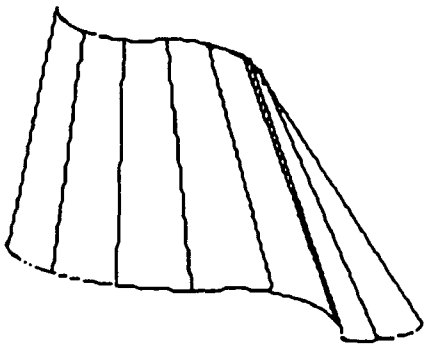


Figure 11: An Example of a RULESURF Mesh. The two curves are at top and bottom of the figure. The "curves" do not have to be formed from arcs; you can use circles, points, lines, polylines, arcs, or 3D polylines.

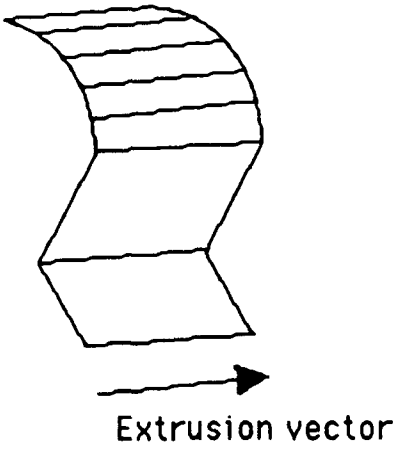


Figure 12: An Example of a TABSURF mesh. The defining curve is at the left side of the figure and is extruded to the right. The drawing plane for the curve is skew to the page and perpendicular to the extrusion vector (which is pointing into the page).

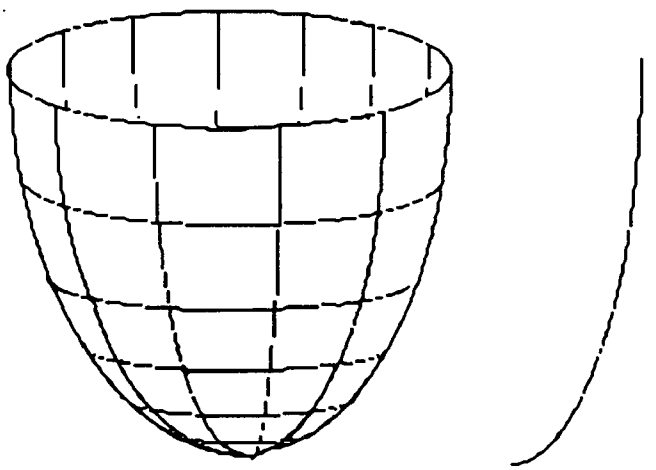


Figure 13: A Surface of Revolution. The curve used to generate the surface has been shifted to the right of the mesh; it was rotated about a line (not visible) inside the surface. The curve at right lies in the drawing plane.

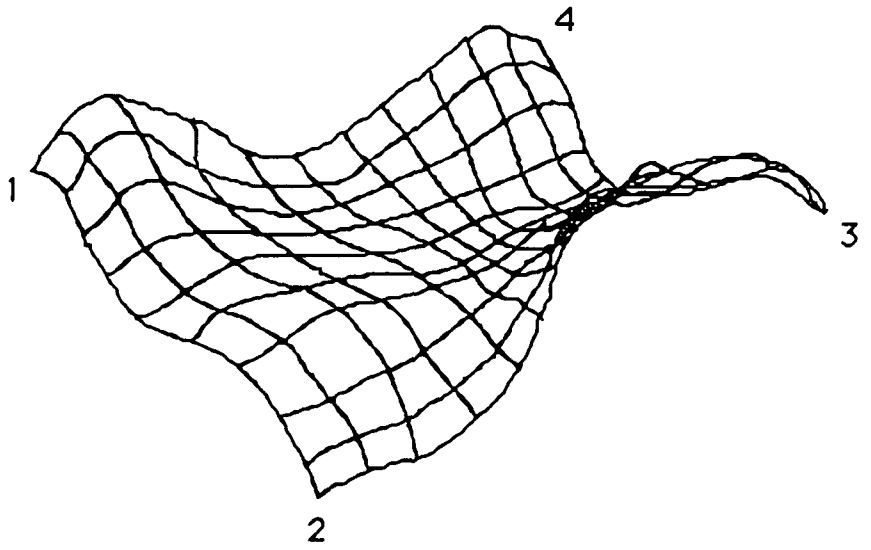


Figure 14: An Example of an EDGESURF. The four defining edges (in this case, 3-dimensional polylines) are from vertex 1 to vertex 2, from vertex 2 to vertex 3, from vertex 3 to vertex 4, and from vertex 4 to vertex 1.

For more details on the use of meshes, see pages 92 - 97 of the reference manual.

1.1.1.7 3-D

There is a special AutoCAD command menu of 3-dimensional shapes and objects called "3-D" accessible from the main AutoCAD menus. It includes the meshes of Section 1.1.1.6 and solids of Section 1.1.1.5, as well as wedges, tori, spheres, domes and bowls. These surfaces are not all well-designed, Spheres and domes are correctly oriented, but the tori and bowl are back-facing. The 3-dimensional pyramid in the script is not 3-dimensional, and so forth. The objects look all right in AutoCAD, but apparently were not designed to be really 3-dimensional. There may also be some more problems in the parser. I have not spent much time getting these objects to work correctly because there has been little need for them, and because all of the special entities can be made with other commands (for example, you can make tori, spheres, domes, and bowls with the REVSURF command). So use the 3-D menu at your own risk; the mesh commands work, but the other commands are not guaranteed.

1.1.2 Degenerate entities

As mentioned throughout Section 1.1.1, the parser does not convert some entities in certain cases. The entities listed here are not converted at all: POINT, LINE (no thickness), PLINE (no thickness), 3DPOLY (3-dimensional polyline), TEXT, SHAPE(a sort of primitive block; very quick to insert, but very primitive). These entities are not 3-dimensional, or are not necessary for 3-dimensional modeling, or are just things no one has gotten around to adding to the parser.

1.2 Entity modification

Entity modification commands fall into two categories: general commands and entity-specific commands. General commands work on all AutoCAD usable entities, and produce uniform -- though sometimes unexpected -- results. The entity-specific commands work on only a few entities, addressing special characteristics of those particular entities.

1.2.1 General commands for entity-modification

There are four important general entity-modification commands: SCALE, STRETCH, MIRROR, and CHANGE. The command names are self-explanatory, except

for the CHANGE command; for example, SCALE changes the size of an object, scaling it in relation to some user-specified point. STRETCH likewise does what it sounds like it should: you can "stretch" an entity by choosing a vertex or several vertices and then moving those vertices with respect to the unselected vertices. These commands do not adversely affect the way the parser interprets the entity; a stretched entity is turned into polygons with the same stretched shape as the entity, but those polygons are not otherwise altered.

MIRROR, on the other hand, alters entities in two ways, one obvious and one subtle. Applying MIRROR to an entity or collection of entities causes them to be reflected about a user-selected line (the user is prompted to enter a pair of points, not an actual line in the drawing), creating a reverse image of those entities. However,.the mirrored entities will be flipped: every polygon in the reversed image will face in the opposite direction from its corresponding polygon in the original collection of entities, as demonstrated in Figure 15. The MIRROR command produces a mirror-image of the appearance of an object, but not of the polygon orientation.

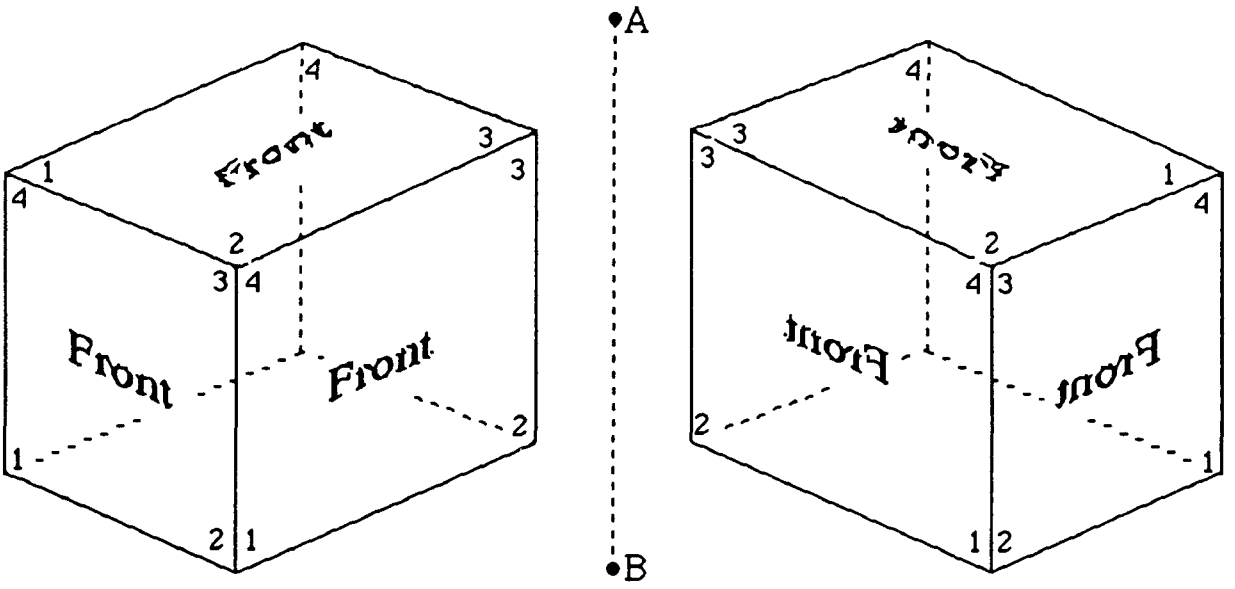


Figure 15: A Box of 3Dfaces Before and After the MIRROR Command. The original box is at left, the mirrored box at right. The points A and B define the line about which the box has been reflected. Polygons making up the original box -- each with vertices ordered <1,2,3,4> -- are reversed in the mirrored copy. The front sides are now on the interior of the box.

CHANGE is a very versatile command, modifying color, thickness, layer, elevation, and line type. CHANGE COLOR changes not only the selected entity's color in AutoCAD but also changes the color of the polygon produced by the parser (and the texture, if used with the Walkthrough texture information). This will be discussed further in Section 4. CHANGE THICKNESS changes an entity's thickness (the distance the entity is extruded), CHANGE LAYER changes the AutoCAD layer the entity is associated with, and CHANGE ELEVATION moves the object along the Z-axis (The CHANGE ELEVATION command is better classified as entity manipulation than as entity modification, since it moves the entity). And finally, CHANGE LINETYPE changes only the representation of the lines used to draw the object in AutoCAD; as far as the parser is concerned, it does nothing at all.

As a supplementary comment, it should be noted that CHANGE does not work on entities drawn in User Coordinate Systems. There is a modified CHANGE command, described on p. 129 of the *AutoCAD Reference Manual*, called CHPROP, that operates on entities even of differing coordinate systems. It operates on a subset of the CHANGE options (it cannot be used to change an entity's elevation), but the most used subset (color, layer, line type, and thickness).

1.2.2 Entity-specific commands

In addition to the foregoing commands, there are, for the line-based entities (line, arc, polyline), additional modification commands. Three that have proven useful in the architectural modeling I have done are the TRIM, EXTEND, and PEDIT commands. TRIM and EXTEND are complementary commands: EXTEND extends a line, arc, or polyline; TRIM clips those entities. In both commands, you select a line, arc, or polyline in the drawing that either should intersect or does intersect the entity you want to extend or trim; it is marked as a boundary. You then select the entity you want to extend or trim, and if it would intersect or if it does intersect the boundary, the entity is is extended or trimmed, respectively. EXTEND is very useful for making walls meet (if you have used lines for walls), and TRIM is useful for clipping away parts of walls that are too long, cutting doors or windows into walls, and so on.

PEDIT, as its name might suggest, is for modifying polylines. There are many ways to modify a polyline with PEDIT; you can move vertices, add vertices, join two

polylines, close a polyline (by connecting the first and last vertices), open a polyline (by disconnecting the first and last vertices), fit curves to the vertices of the polyline, change the width of a polyline, or undo your last edit. It can also be used to modify polygon meshes, though for a large mesh it can be very tedious to edit all the vertices. However, if needed, the command is there.

The EXTEND and TRIM commands are transparent to the Walkthrough parser; if given a trimmed line and an equivalent untrimmed line as input, the parser will convert them to equivalent polygons. PEDIT is equally transparent, although the parser does not handle all PEDIT options (for example, the parser does not fit curves to points, though the AutoCAD PEDIT command does).

1.3 Entity manipulation

The basic AutoCAD entity manipulation commands, MOVE, COPY, ROTATE, CHANGE ELEVATION, and ERASE, perform the tasks of moving, copying, changing the elevation, and erasing entities. These commands are absolutely transparent to the parser; manipulating an AutoCAD entity will manipulate the polygons approximating it in the same way. More details on the use of these commands may be found in the *AutoCAD Reference Manual.*

2 Blocks and Layers

As your drawing becomes more complex, you will find that although one entity is easy to manipulate, collections of entities are less so. It is trivial to manipulate a floor, but somewhat difficult to manipulate an entire building. It is also tedious to copy whole groups of entities one-by-one, especially if you want to modify or manipulate the copies.

AutoCAD has two techniques to help manage the database: BLOCKs and LAYERs. A BLOCK is a collection of primitives that can be manipulated as a single entity; for example, one might want to have a BLOCK called **chair** to place in a model of a restaurant, rather than 50 copies of the polygons constituting the chair. The LAYERs help in a different way; entities (and BLOCKs) can be assigned to different layers that represent different aspects of the model. For example, in a building model, the

furniture might be put in a single layer, the first floor in another layer, and so on. Undesired LAYER can be frozen (using the LAYER command and FREEZE to turn off layers); the frozen layer is not included in the display list, speeding up REGEN commands, and not interpreted as part of the DXF file by the parser, creating a subset of the database.

2.1 Block creation

Blocks can be created in either of two ways: explicitly within a drawing or implicitly by inserting an existing drawing. The BLOCK command lets you pick a group of entities, give them a name and an origin, and then allows you to treat them as a single entity. The origin becomes the center of the block, used to position the block in the drawing (INSERT allows you to rotate the block about its origin after insertion). The block, once defined, disappears from the drawing (the individual entities vanish, and are not replaced by a copy of the block. You must INSERT the block to reposition it within the drawing). Blocks may be treated like entities and assigned an elevation, color, and layer (if there are default elevation, color, or layers set, the block will assume those values).

Blocks may also be made from existing drawings; if the INSERT command is given the name of a .dwg file, that drawing will be inserted as a block with its origin as its positioning point. Models can thus be modularly constructed; you can build up parts of a model at a time, building small objects such as furniture or windows in separate files, and then inserting them into a larger model. This serves a number of purposes. The parts can be modified and updated once, in the smaller file, and updated with

INSERT <old_block>=<new_block>

For example, to replace the block named "grand_piano" with the model in piano.dwg, you would type:

INSERT grand_piano=piano

You should replace <old_block> with the block name and <new_block> with the file name (without the file extension). Do not type any spaces about the "=" because AutoCAD will interpret the spaces as carriage returns. This form of the INSERT

command will replace the copies of the old block with the new, although they may not appear that way in AutoCAD immediately (AutoCAD doesn't always update its display list immediately).

The advantage in inserting files as blocks is twofold: you can save time by reusing existing blocks, and it makes the drawings easier to manipulate. Parts can be re-used; elements of one drawing can be put into a different drawing, saving drawing effort. Secondly, it is far easier to find and manipulate a single entity than it is to find and manipulate a dozen, especially in a large and complex drawing. It is convenient to build small parts of a design in separate drawings, and insert the pieces as blocks. Hierarchical models can be built from hierarchical blocks, and be manipulated as a single object.

The INSERT command has several options: insertion point, scale, rotation, and attributes. Obviously, the insertion point is where in the drawing the block should be inserted; AutoCAD will accept a point from the keyboard or a point selected with the cursor in either 2 or 3 dimensions. The scale options allow you to scale the block along the X, and Y, and -- if specially requested with the XYZ option -- Z axes; this can be convenient for bringing small blocks to scale in a drawing, but can be a grand pitfall if you ever need to modify the block, as will be covered under EXPLODE in the block modification section below. You can scale the block uniformly, by setting each scale factor the same, or you can scale differently along different axes. Blocks can be rotated about the insertion point, either under control of the cursor or by entering an angle at the keyboard. Lastly, there are the attributes, if present, that are defined when the block is inserted. Attributes are a more complicated item than really should be covered here; however, they are very useful for assigning special information to a block. For example, Walkthrough assigns lighting and texture information to special lighting and texture blocks. The parser can process only very specific types and formats of attributes, however, so their use is somewhat limited.

2.2 Block modification

Blocks can be modified similarly to the way entities can, with limitations. You cannot use TRIM, EXTEND, or PEDIT on a block, even if the block is made up of entities that can be trimmed, extended, or "pedited." Nor can you STRETCH a block; if you must STRETCH a block, you will need to try reinserting it with different scale factors to

get the effect. You can SCALE and CHANGE a block; SCALE functions the same as with entities but CHANGE works a little differently. If the entities that make up the block were given a color, then that color cannot be changed (except by changing the entity and then reinserting the block). If, on the other hand, the entities were not given a color or were given the color BYBLOCK, then whatever color is assigned to the block is assigned to the entities. You can even make blocks of some colored entities and some BYBLOCK colored entities; this has been used, for example, for wooden chairs with colored cushions, sinks for differently colored bathrooms (but all with the same metal fixtures), and lamps with a set shade color but a varying color for the base.

It is possible to turn a block back into its component entities with the EXPLODE command. An "exploded" block is replaced by the individual polygons that made up the block and the information assigned to the block (e.g., color) is lost. You can then STRETCH, TRIM, EXTEND, PEDIT, etc., the entities as much as you like. There are limitations; a block that was not scaled equally in all directions when inserted cannot be exploded, so if you scaled X by 5 and Y by 2, you cannot explode the block. Also, it is not possible to turn the exploded entities back into the original block, since you may have changed the entities (you can UNDO the EXPLODE command).

2.3 Block manipulation

Block manipulation is much more like entity manipulation than block modification is like entity modification. You can MOVE, COPY, CHANGE (with the limitations noted in Section 2.2), ROTATE, MIRROR, and ERASE blocks in the same way you moved, copied, changed, rotated, mirrored, and erased entities; the commands function exactly the same.

3 Advanced Modelling

Advanced AutoCAD modeling is dominated by the need to manage large and complex models. The modeling techniques and commands do not change; however, the model sizes do. Dramatically. Realistic models tend to be very complex and frequently large both in polygon count and in area; Walkthrough typically works with models of around 5000 - 8000 polygons representing buildings that are between 50 to 150 feet long on a side. For example, compare Figure 16: A Simple Church Hall with Figure

17: A Plot of the Orange United Methodist Church Model. Both model a building that is about 90 x 100 feet, but Figure 16 uses only 7 polygons whereas Figure 17 uses approximately 7500. Clearly, Figure 17 is the much more complicated drawing, and when compared with the real Orange United Methodist Church Fellowship Hall, the more accurate one. However, this complexity also makes it a more difficult file to work with.

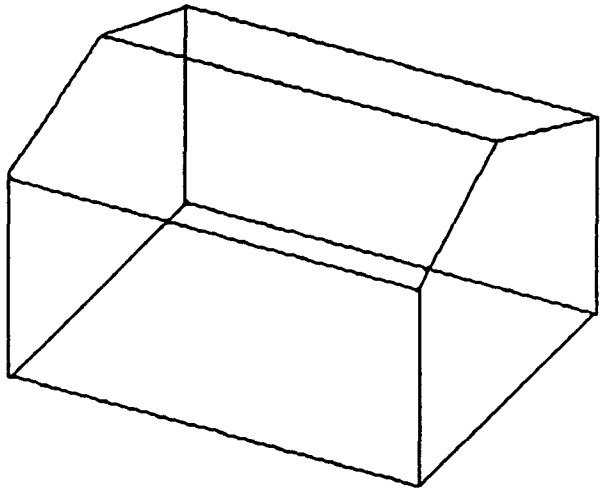


Figure 16: A Simple Church Hall

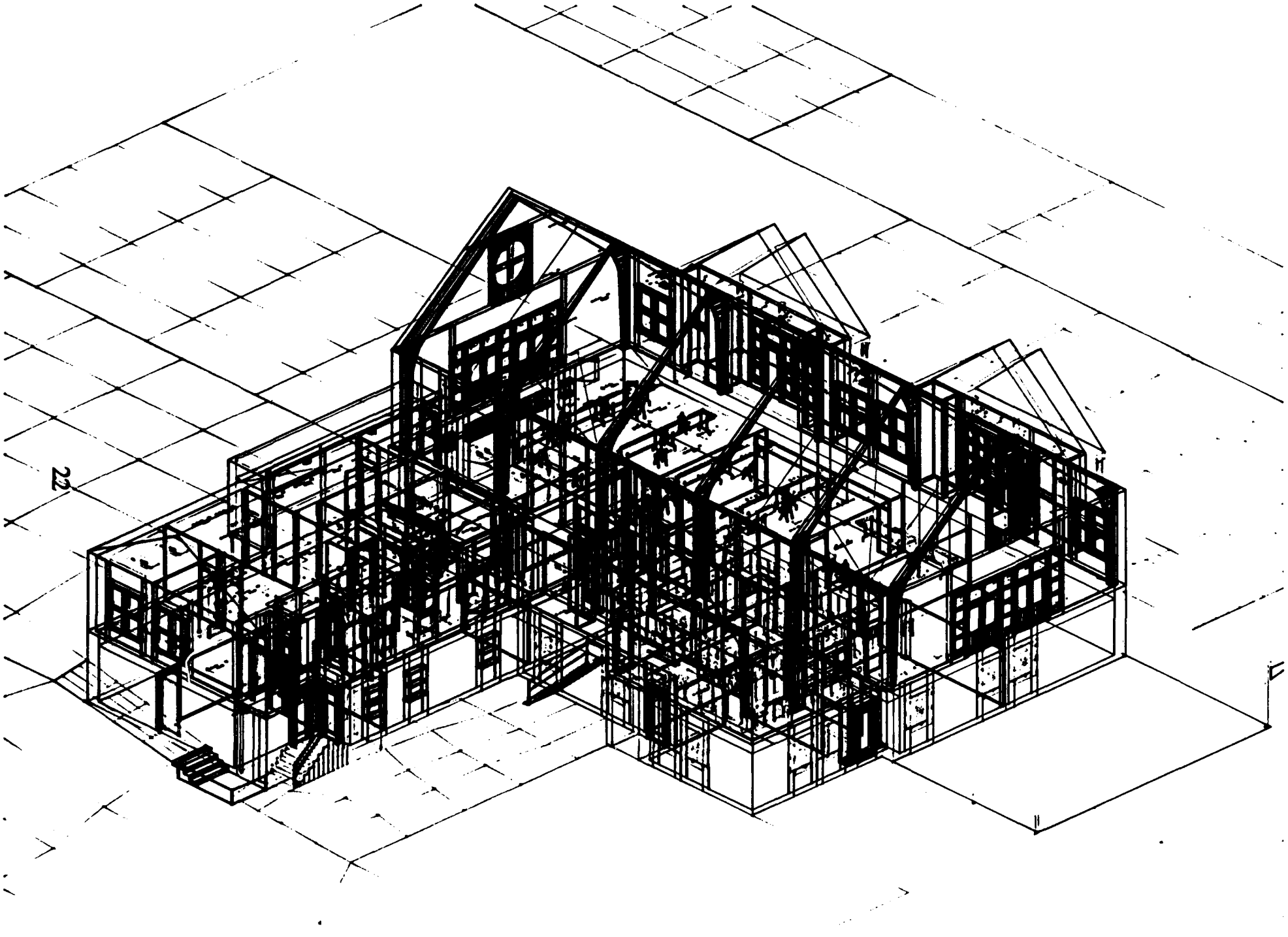


Figure 17: A Plot of the Orange United Methodist Church Model

It is not difficult to change a single polygon in a simple model of 100 or fewer polygons; the model is small enough that you can keep track of its underlying structure and how the polygons in the model are related. In a complex model, this kind of knowledge of the structure is not possible: there are too many polygons, and the model represents too large an area. If we wanted to change the color of one polygon of Figure 17 -- granted that we could find it in such a view -- we would have a lot of difficulty picking the one polygon from the thousands on screen. You can't even tell which polygons belong to the upstairs and which belong to the basement. Also, it becomes difficult to get any sort of accuracy in picking points in the model -- much less figuring out what is what -- unless you zoom in on a very small section. Some form of order is needed to manage the polygon chaos, and although it is unlikely we will be able to reduce the total number of polygons in the model (and still maintain the same degree of realism), we can reduce the amount of complexity. By using blocks and layers, we can impose order on the model.

3.1 Blocks

Blocks are very useful in large models. Built separately from the main drawing, a block is generally a small set of polygons in a relatively simple figure, something that you can fit on the screen at a reasonable drawing scale. The main drawing -- the one the block will go in -- may be 100 feet on a side, which works out to about 10 feet in the model to one inch in the drawing window; it becomes very difficult to work accurately under these conditions. In the block, the scale tends to be less than 1 drawing-scale foot to the inch on the window. It is far easier to work in this scale.

It is also easier to manipulate the drawing of the block; with fewer polygons, it takes less time to regenerate a small drawing, to load a small drawing, to save a small drawing.

The best reasons for using blocks, however, have nothing to do with either of the above points; these are that blocks are reusable entities -- objects that can be moved, changed, and rotated as a single entity, inserted at will in any drawing. If you want to rearrange the furniture in a room. you need only move the chairs, the pianos, and the tables, not the polygons that make them up. Selecting any polygon in the block selects the block; you need not hunt for each polygon making up the chair. If you decide you

want a different chair, or a corrected version of the old chair, an INSERT <block>=<new block> will replace all instances of <block> with <new_block>; you need not erase every polygon of every chair. Creating a model becomes a quicker task: any repeated object -- door, window, chair, table, light fixture -- can be a block and need be drawn only once. Blocks can be inserted within blocks; a chair can be inserted in a house, the house in a neighborhood, the neighborhood in a city, and so on.

By using blocks, you make the drawing more modular and end up, consequently, with fewer entities that can be individually changed. There are still the same number of polygons in the drawing, of course; however, they are at least clustered. A further reduction in complexity can be made with the use of layers to order the drawing.

3.2 Layers

The advantages of using layers are twofold: you can reduce the number of things you see on the screen, and you can reduce the number of polygons produced by the Walkthrough parser. When you freeze a layer, you turn it off, in effect: it is not drawn, and it is not even considered to be be part of the display list. Regenerating the screen is faster, since the entities in the layer are not considered to be part of the displayable drawing. Polygons that otherwise would be in the way are not displayed. So by putting the entities into different layers, you can control which parts of the drawing you will be working with: you can turn off the ones that will not be needed, working only with those sections that will be.

The LAYER command offers several options: ?, MAKE, SET, NEW, OFF, ON, COLOR, LINETYPE, FREEZE, and THAW. In AutoCAD, there is always a current layer. When an entity is drawn, it is assigned to whichever layer is current at the time of its creation. The entity's layer can be subsequently changed, but it is initially the current one. You can make a layer current with LAYER SET. You can create a new layer and make it the current layer by using LAYER MAKE, or by using LAYER NEW to create the layer and SET to make it current (the name of the current layer is displayed in the upper left-hand corner of the drawing at all times). LAYER ? displays a list of all existing layers. Layers can be turned off and on, or frozen and thawed. ON and THAW undo OFF and FREEZE; layers can only be turned on or thawed if they have been previously turned off or frozen. Layers turned off or frozen are not shown. Frozen layers are completely removed from the display list, making drawing regeneration times faster;

off layers are kept in the display list and culled at the last step, after transformations but before rendering.

The parser recognizes frozen layers, but does not distinguish off layers from ordinary layers. Layers frozen before a DXFOUT will be not be turned into polygons; all other layers (including off layers) will be transformed into polygons. This can be used to transform subsets of larger models; if you want a certain subset of the drawing, put the undesired entities into frozen layers and do a DXFOUT. The file produced from that DXF file will not include anything from the frozen layers.

4 Converting from AutoCAD Form to Polygons

Conversion from the AutoCAD format to the form I will refer to as a *poly* file is done with the Walkthrough AutoCAD-to-polygon parser, a program called acadtopoly. It accepts a .dxf file (so you will need to do a DXFOUT before running it) and a color-table file (.ctab file) as input and writes the polygons to the standard output. The .dxf file is ASCII representation of the model and can be treated as a black box: it is not necessary for the user to know anything about the contents of the file to convert it to polygons (Interested users can find out more about the format in Appendix C of the *AutoCAD Reference Manual*). On the other hand, the .ctab file, the parser, and the .poly file may require some user-interaction or modifications, and will be individually discussed.

4.1 Color-table

A color-table file has the form

x r g b (t)

where x is the index in the file, r g b is an rgb (red green blue) triplet, and the optional t is a texture number. The index is the same as the color numbers assigned in AutoCAD: the AutoCAD color is mapped to the color represented by the rgb triplet. AutoCAD allows up to 255 colors, so you can have up to 255 colors in your model. Producing the colors for the color table is not as easy as in AutoCAD; the rgb triplet as a way to specify a color is a difficult concept to grasp. Knowing that black is 0 0 0 and

white is 255 255 255 does not make finding brown intuitive. So, there are interactive tools to help, such as the program "colortool" on pxpl4, which let you change the screen color by moving sliders to manipulate the red, green, and blue values (the control panel for the 256-color Macintosh monitor has a similar kind of tool). There is also a default color file for acadtopoly, called acad.ctab, if you do not want to have to make your own; it assigns the colors used in AutoCAD to the polygons.

4.2 Running acadtopoly

You will need to be on a DECstation (3100, 5000, or anything with a MIPS chip) to run acadtopoly. The program is called acadtopoly*X*, where *X* indicates the type of machine acadtopoly is compiled for. Currently, there is only a version for the DECstations, so it is called acadtopolyM ("M" for "MIPS").

The acadtopoly executables are kept in /louvre/glab1/walk/bin, so you will be running /louvre/glab1/walk/bin/acadtopolyM. Running acadtopolyM without any arguments will give you instructions on how to run the program, and acadtopolyM -\? will list of all the possible arguments. If you want to include texture information, you will need the -T option, and if you have lights you should use the -A option, but the rest are not necessary for typical models.

The command format is

```
acadtopolyM -options dxf_file.dxf ctab_file.ctab > output.poly
```

You should include the file extensions .dxf and .ctab if you are providing your own dxf and ctab files; acadtopoly looks in default directories for its inputs. The .dxf and .ctab file extensions indicate it should use the files in the current directory rather than the default. If you want to use the default .ctab file, acad.ctab, you should type "acad" for the ctab_file name in the command.

4.3 Output

The output of acadtopoly is a file in one of two formats. If you are familiar with texturing, have included texture information in your color-table file, and use the -T option, the output will have polygons with a 10-field header:

```
fr fg fb rr rg rb n_vert emit t_id t_index
x1 y1 z1
x2 y2 z2
.....
xn yn zn
```

where

- fr, fg, and fb are the red, green, and blue components of the front face color
- rr, rg, and rb are the red, green, and blue components of the back face color. The parser automatically sets back face color to 255 0 0 (red), to make incorrectly oriented polygons easier to find.
- n_vert is the number of vertices for this polygon
- emit is the emitter id field (zero if this is not a Walkthrough emitter)
- t_id is the texture id field for this color-table entry
- t_index is the index into a file called <dxf_file>.texture (<dxf_file> is the name of your .dxf file from the acadtopoly command of Section 4.2), containing orientation information for the texture.

The x1, y1, z1 through xn, yn, zn triplets are the vertices that make up the polygon; the "n" in xn, yn, and zn equals n_vert.

If you are not familiar with texturing and do not want texture data, do not use the -T option and the format will not have either the t_id or t_index fields. Your output file will have an eight-field header:

```
fr  fg  fb  rr  rg  rb  n_vert emit
x1  y1  z1
x2  y2  z2
.....
xn  yn  zn
```

where the abbreviations are the same as for the 10-field header (if you are interested in adding textures, see /louvre/glab1/walk/doc/texture.ps for more information).

Either format can be further modified, using awk scripts or programs written specifically for converting poly format to other forms.

4.4 Correcting the Polygon Model

If you have built your model carefully and with attention to the orientation of the entities, then the polygon model should have all its polygons correctly oriented and positioned. This is a difficult task, so it frequently the case that there are backfacing, coincident, or misoriented polygons in the model, which must be corrected. There are some tools (of varying degrees of success and robustness) to help you fix some of these problems; we will discuss the problems and the tools that correct them.

4.4.1 Backfacing polygons

Backfacing polygons are the most common problem, and the easiest to fix. There is a Walkthrough program that runs on Pixel Planes 5, called tool, which lets you steer through the model with joysticks and zap backfacing polygons (which are colored red). The command format is

```
/louvre/glab1/walk/bin/tool  -P  poly_file
```

where poly_file is the name of a poly file. It runs on Pixel Planes 5, using the high resolution monitor (there is a cross-hair rendered in the center of the monitor; the detail of the cross-hair requires the high resolution monitor).

When you start tool, it loads the polys and converts them to pphigs polygons for display. Depending on the size of the model, this may take several minutes (a "." is printed every 1000 polygons to indicate the loading is going well). The Help screen is printed after the initialization completes. You can now fly about the model and flip backfacing polygons. The left joystick controls translations, the right controls rotations (except for head roll, which is disabled).

A complete tool tutorial is beyond the scope of this guide. However, these are the basic commands for correcting backfacing polygons:

- "h" prints the help screen
- "q" quits tool
- "w" writes the altered model to "<file>.flipped.poly"

- "f" flips the polygon under the crosshairs
- "X" deletes the polygon under the crosshairs
- "u" undeletes the last deleted polygon
- "c" undeletes the next-to-last deleted polygon
- "F" toggles between joystick control of the crosshairs and joystick control of the view. The default mode is to use the joysticks to fly the model under a fixed crosshairs; "F" makes the joysticks control the movement of the crosshairs over a fixed view of the model.

A polygon can be flipped indefinitely; if you accidently flip the wrong polygon, you can flip it again and correct the mistake. tool does not automatically write out the altered model; you must explicitly select the write command before quitting or you will lose your corrections.

You can convert the corrected model back into a dxf file using polytodxf. The command format is similar to acadtopoly:

```
/louvre/glab1/walk/bin/polytodxfM  poly_file  ctab_file  >  output_file.dxf
```

where poly_file is the name of a polygon file (without the .poly extension), ctab_file is the name of a color-table file (without the .ctab extension), and output_file.dxf is the name of the output file.

Be forewarned: because the poly format is not as complex as the dxf format, the dxf file produced by polytodxf will not contain as much information as the original dxf file. **You will lose all layer and block information.** The new dxf file will contain no blocks, only entities, and all in a single layer (layer 0). If you are working on a simple model this will not be a problem and you need not worry about it. If you are working on a complex model, like the Orange United Methodist Church Model in Figure 17, then you should try to correct the blocks before insertion into the larger model, and correct large models a layer at a time. You can FREEZE all but one layer, do a DXFOUT, convert to polygons (acadtopoly ignores frozen layers), flip the backfacing polygons using tool, convert back to dxf format, ERASE all the polygons of the layer (to avoid copying the corrected polygons in on top of the old ones), DXFIN the layer, and then CHANGE LAYER on the corrected polygons to put them back in their original layer (since polytodxf put all the polygons in layer 0).

4.4.2 Coincident polygons

Coincident polygons are polygons that share the same space, overlapping at some point. They are more difficult to find and correct than back-facing polygons. If the polygons are different colors, then finding the coincident polygons is simple and can be done with tool. It will be obvious, actually, since the polygons will appear to "flash" as parts of one are rendered, then parts of the other. Because they occupy the same space, the Z-buffer will not consistently display one or the other, but will swap between the two from frame to frame. You can remove one either in AutoCAD, or with tool (instead of using the "f" key to flip the polygon, use the "X" key to delete it).

Unfortunately, this is the rare case. The collective experience of the Walkthrough project models teach that it is much more likely that you will have coincident polygons of the same color, and that there will be no obvious flashing between the polygons (the polygons are the same color, so you cannot see the flashing). There is currently no good software solution to prevent this problem. If you calculate a radiosity solution for the model, then you will find the coincident polygons flashing when you display the solution (one polygon gets most of the light, the other only a little). In the Walkthrough project, we go back to the model and correct it at this point, older but wiser.

4.4.3 Misoriented polygons

This term covers a multitude of sins, and a multitude of heinously difficult problems. A misoriented polygon can be one which gets lost in the modeling pipeline (a face modeled with a polyline or a non-planar polygon, for example) or a entity drawn at the wrong elevation which appears to be missing but is actually elsewhere (coincident or inside of an object), or a polygon which, through modeling error or round-off error, is slightly and almost imperceptibly shifted. It is easy to verify that a polygon is not in its expected position with tool or some other display program. It is more difficult to tell if the polygon was lost in the parser, drawn in the wrong place, or never existed. If the polygon was drawn or shifted to the wrong place it may come up later, in an unexpected and inappropriate place. The very difficult case is to detect and correct the fractionally misaligned polygon. The misaligned polygon will show up as a shading discontinuity in a radiosity model or a small flash as we see between two

apparently connected polygons. There are currently no software solutions at UNC to correct these problems. Fortunately, these problems tend to be relatively rare. Unfortunately, they tend to be appallingly obvious when found and thus difficult to conceal.

5 Conclusion

The Walkthrough AutoCAD parser interprets most AutoCAD commands and entities as the AutoCAD program does; the polygons tend to come out of the parser looking as they did in AutoCAD, with the exceptions noted in the text. Object manipulations and modifications are transparent to the parser, with the exception of the MIRROR command. In short, a model made in AutoCAD will produce a polygon model of similar appearance.

With an appropriate amount of care, AutoCAD can be a tool for accurate model building; 3-dimensional drawings made with the above information in mind will produce accurate, 3-dimensional models. Whether this sort of treatment is appropriate for a model is a decision each user must make for his or her particular design. This guide should help with the decision whether AutoCAD is the appropriate modeling tool for the job.

BIBLIOGRAPHY

AutoCAD Release 10 Reference Manual. Publication TD106-001.1. Autodesk, Inc. Sausalito, CA. November 4, 1988.

AutoCAD Release 10 Installation and Performance Guide: Edition for the Sun Microsystems computer running the UNIX operating system. Publication TSU3IG003.00. Autodesk, Inc. Sausalito, CA. December 8, 1988.

Rheingans, Penny and John Alspaugh. *The Orange United Methodist Church Fellowship Hall.* Drawn in AutoCAD from plans by McClure-NBBJ of Raleigh, NC. Department of Computer Science, University of North Carolina at Chapel Hill, 1989.

An Environment-Projection Approach to Radiosity for Mesh-Connected Computers

Amitabh Varshney **Jan F. Prins**

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175
USA
varshney@cs.unc.edu prins@cs.unc.edu

Abstract

We describe a progressive refinement radiosity algorithm for highly-parallel mesh-connected SIMD or MIMD computers. The technique is based on environment-projection and scales easily to large machines and datasets. Form-factor computations can be performed using local communication by mapping the single-plane across the processor mesh. We report on the performance of an implementation on the MasPar MP-1, and discuss some potential improvements related to load balancing.

1 Introduction

In recent years, the radiosity method has gained widespread acceptance in the computation of view-independent global illumination for (primarily) architectural datasets. One of its important applications has been to provide realistic lighting for datasets of proposed buildings generated from architectural drawings. By navigating through one of these virtual buildings, the architectural design can be evaluated for usability, traffic, and aesthetic appeal [Airey90b]. Any necessary modifications, to remove shortcomings or to test new ideas, can be made and the design re-evaluated. One of the major bottlenecks in this design cycle is the computation-intensive radiosity process. With the increasing availability of parallel computers, attempts are being made to parallelize this process to achieve interactive rates on non-trivial datasets (of the order of thousands of polygons), thereby making the whole design process a more meaningful one.

In this paper we describe a radiosity algorithm for highly-parallel mesh-connected computers that can provide rapid progressive estimations of global illumination of large datasets. The organization of the remainder of this paper is as follows. We start with a brief overview of radiosity that introduces the terminology we will be using. Next we describe the parallelism available in the computation of radiosity and classify previous work done in parallel radiosity. Next we describe our approach and its implementation on a 4K processor MasPar MP-1. We conclude with a discussion of (as-yet unimplemented) potential improvements of our approach.

2 Radiosity Overview

Radiosity B_j for a surface j is defined as the total rate at which radiant energy leaves that surface in terms of energy per unit time and per unit area [Goral84]. In an environment composed of n diffuse surfaces the radiosity of a surface j is given by

$$B_j = E_j + \rho_j \sum_{i=1}^{n} B_i F_{ji} \tag{1}$$

where E_j is the rate of direct energy emission from the surface j per unit time and per unit area, and ρ_j is the reflectivity of the surface j. The summation in the equation 1 represents the total flux incident on surface j from all other surfaces in the environment. F_{ij} is the *form-factor* for surface i with respect to surface j denoting the fraction of radiant energy leaving surface i that is incident on surface j.

Considering all n surfaces, this yields a system of n linear equations in the n unknowns $B_1, .., B_n$ shown in Figure 1. The F_{ij} are determined from the geometry of the environment and the E_j, the emittances of the light sources in the environment, are assumed given.

$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \dots & 1-\rho_n F_{nn} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix}$$

Figure 1: The Radiosity System of Equations

The form-factor F_{ij} is equal to the fraction of the base of a unit-radius hemisphere centered on surface i that is covered by the projection of surface j on that hemisphere. The *projection-surface*, whether a hemisphere or its approximation - the hemi-cube [Cohen85], is subdivided into smaller elements, say pixels, for the purposes of sampling the environment. The *delta form-factor* Δf_q for a pixel q on the projection-surface is defined as the contribution of that pixel to the form-factor value. The magnitude of the contribution depends on pixel location. The total form-factor F_{ij} for surface i with respect to the surface j is the sum of all delta form-factors that are covered by the visible portion of surface j on the projection-surface.

Since the coefficient matrix is diagonally dominant[1], iterative methods for solving the system of equations in Figure 1, are preferred over direct methods such as the Gaussian elimination.

In an iterative method, a given variable B_i is updated based on the current approximation to the solution. This corresponds to a "gathering" of all the energy incident on the surface i. The method of progressive refinement [Cohen88] is a variant of the Gauss-Seidel iterative method that computes an update to *all* variables based on the current approximation to B_i. This corresponds to "shooting" all the energy from surface i. By choosing surfaces with high emittance first, this method provides good approximations for radiosity of all surfaces at early stages.

[1] the participating surfaces are assumed to be planar so that $F_{ii} = 0$ for all i

3 Parallel Approaches and Classification of Previous Work

The radiosity problem contains many opportunities to employ parallelism in the computation of its solution.

First, at the level of updating the approximate solution, we may employ a method other than the sequential Gauss-Seidel method of successive displacements. A simultaneous displacement method for solving the radiosity system of equations, such as the Jacobi method, permits all variables' updates to be computed in parallel. In a hybrid method a set of k out of a total of n variables can be chosen as the variables to be solved for in the current iteration. Each of these k variables is solved for in parallel using the values of the previous iteration for all variables. Then in the next iteration, these new values of k variables are simultaneously used.

To perform an update of the solution using surface i, either in the shooting or in the gathering approach, all form-factors F_{ij} are needed and may be computed in parallel. Since form-factor F_{ij} is in turn the sum of those delta form-factors where j is visible, this computation may also be performed in parallel for all j.

Figure 2 summarizes the parallelism available in the problem. The levels of parallelism are nested. One can go down these levels with increasing availability of processors to exploit increasing parallelism.

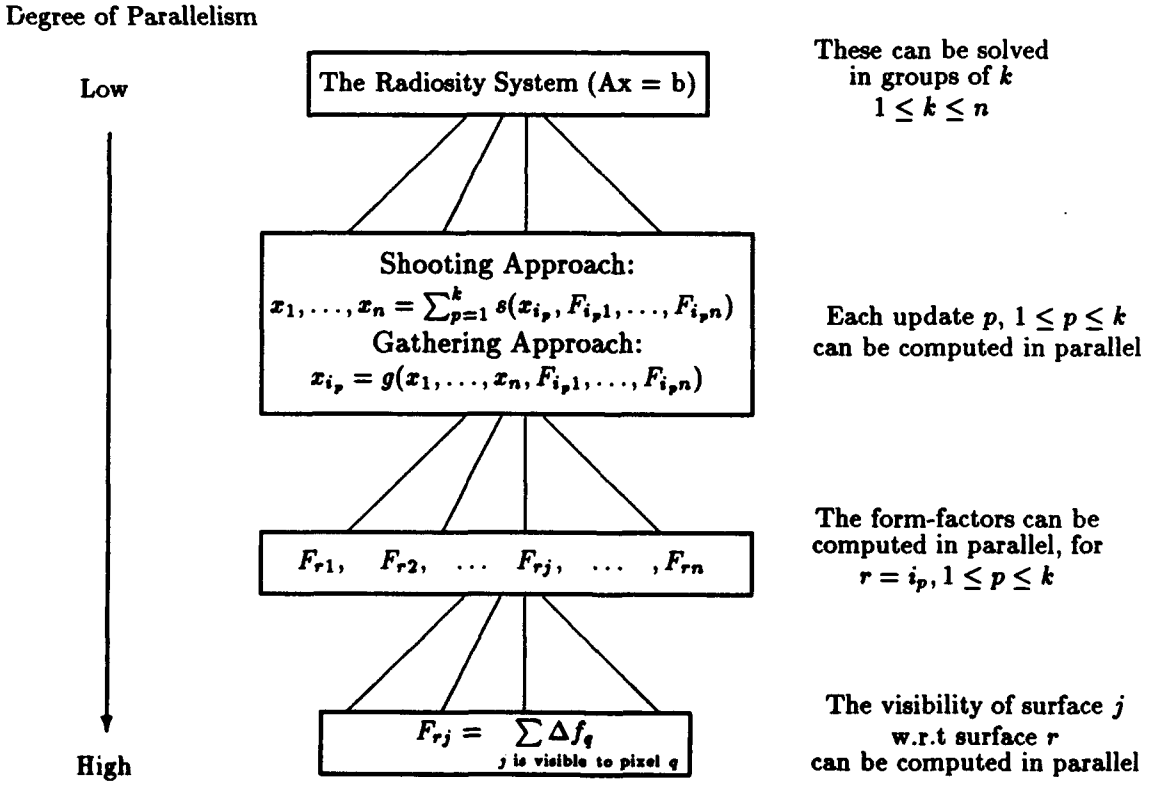


Figure 2: Parallelism in Radiosity

The existing work in parallel radiosity can be classified into three non-disjoint groups based on the level at which parallelism is exploited.

- **Level 1 Parallelism:** This involves computing several updates in parallel, each using one row of the form-factor matrix. This group includes [Chen89], [Chalmers91], [Feda91], and [Guitton91].
- **Level 2 Parallelism:** Form-factors within a row are calculated in parallel. [Baum90], [Gifford91] fall in this group.
- **Level 3 Parallelism:** Delta form-factors are computed in parallel. Baum and Winget [Baum90] achieve this by using the parallel rasterization hardware on the Silicon Graphics workstations.

The level at which radiosity is parallelized in these implementations is based on the extent to which a particular level of parallelization fits the underlying architecture.

Another architecture-dependent design decision is the computation of F_{ij}. In a *ray-casting* approach, rays are fired through points on the hemi-cube to determine the visible surfaces. Energy carried by a ray fired from surface i through pixel q on the hemisphere and reaching a visible surface j contributes Δf_q to F_{ij} [Airey89], [Guitton91]. Alternatively, by firing a number of rays from surface i directly to sample points on surface j, the delta form-factors for successful rays may be added to give F_{ij} [Gifford91], [Hanrahan91].

The *environment-projection* approach involves projection of all surfaces in the environment onto the hemisphere surface or an approximation of it (the hemi-cube or a single-plane), and determining the visible surfaces at each pixel. The form-factor F_{ij} is the sum of all delta form-factors Δf_q where surface j is visible [Airey89], [Chen89], [Baum90], [Chalmers91], [Feda91]. The determination of surface visibility at a particular pixel is done by depth-buffering the surfaces whose projections cover that pixel. The visible surface-id at each pixel is stored in an *item-buffer* for that pixel.

Unlike a ray-casting approach, where the sampling is done *from* the surface i, the environment-projection approach samples the environment *to* the surface i. This guarantees that within an error tolerance equal to the resolution of the projection-plane, no surfaces will be missed in form-factor calculations.

Our objective in this work was to choose an approach for a highly-parallel mesh-connected SIMD computer. In the ray-casting approach, the critical component is the efficiency of the ray-polygon intersection technique. An exhaustive approach that intersects a ray with all polygons works very well [Gifford91], [Varshney91], but is limited to small datasets. [Delany88] proposes a more efficient space-subdivision ray-tracing method, but it has high cost and limited applicability in our setting. Consequently, we investigated an environment-projection approach. By mapping the projection-surface onto the processor mesh we can exploit coherence in the projection of a polygon onto the projection-surface. We use a shooting method as opposed to a gathering method to get progressive approximations for the entire scene.

Our current implementation parallelizes the radiosity computation at the second and third levels described in Figure 2, but could be extended to obtain concurrency at all levels.

4 MasPar MP-1 Overview

Even though the algorithm described in this paper is applicable to all mesh-connected computers, its implementation was carried out on the MasPar MP-1. This section provides a brief overview of the MasPar MP-1 before describing the implementation. A more detailed description of the MasPar MP-1 can be found in [Maspar90].

The MasPar MP-1 is a SIMD computer with a scalar execution unit and a data-parallel unit (DPU) which is an array of processing elements (PEs). Each PE has access to 16KB of local memory and can perform floating-point operations at a rate of about 75KFlops.

The PEs are organized in a 2D mesh with toroidal wrap-around on all edges. Each PE has direct connections to its 8 nearest neighbors and these are used for providing fast local communication (**xnet**). Relatively slower global communication is provided by a separate multistage crossbar network (**router**). This provides the lower-bandwidth general communication between arbitrary PEs.

The machine on which the implementation was carried out has 4096 PEs.

5 The Algorithm

The polygons in the model are subdivided into patches and spread out equally over the PE-array. Each patch has a *shooting energy* which initially equals its emittance. The algorithm consists of repeatedly performing the following five phases of calculations until the radiosity of the patches has been approximated to within a predefined threshold.

Shooting patch selection phase

Since the progressive refinement approach is used, at each iteration we simultaneously distribute the energy from the k patches with highest energies. In the current implementation $k = 1$, yielding a single *shooting patch* i that is easily found using a global parallel **reduceMax** operation over the shooting energies of all the patches.

Projection phase

During an initialization phase, a matrix that transforms the world-coordinate system to the patch's projection-surface is computed and stored with every patch. The transformation matrix stored with a shooting patch is used by all the other patches to compute their projection in parallel onto the projection-surface of the shooting patch. The projection-surface in our implementation is a single plane that is able to catch 90% of the light energy emanating from the energy shooting patch. The idea of approximating a *hemi-cube* by a single-plane has been described in [Recker90].

The single-plane is mapped across the processor mesh to maintain an orthogonal and monotonic correspondence between x and y on the plane and the processor indices such that the single-plane covers the entire mesh. Thus, neighboring pixels on the single-plane fall onto either the same processor (if the resolution of the single-plane is greater than the number of processors available) or onto an immediately neighboring processor. This mapping was chosen to exploit the coherency expected among the nearby pixels on the single-plane.

Each transformed patch determines its upper left corner on the single-plane and sends a description of the patch to the corresponding processor on the mesh using the **router** network.

Scan-conversion phase

Patch descriptions are spread-out downwards on the single-plane through the **xnet** to give single-processor thick strips. All strips (each corresponding to one patch) are then spread-out to the right in parallel, again using the **xnet**. Hence z-buffering at a processor is a matter of finding the patch description with minimum z value in the processor. The pixels that fall within the bounding-box but outside of the actual projection are not considered for z-buffering. To conserve space and improve performance, accumulation of the patch-descriptions is done only during the downward spread. During the horizontal spread, z-buffering is done on the fly as the patch descriptions travel across

the processor mesh.

Form-factor calculation phase

Once the z-buffering is done, a surviving description of patch j at a pixel q on the single-plane must add Δf_q to F_{ij}. This is accomplished with a send to patch j of Δf_q using addition as a combining function. This operation is implemented efficiently, and in a way that is independent of the distribution of patches on the single-plane, using a single sort operation on the mesh followed by parallel-prefix sum and a send on the **router** network.

Radiosity update phase

The shooting energy of patch i now updates the radiosity of each patch j using

$$B_j = B_j + B_i \rho_j F_{ij} A_i / A_j \tag{2}$$

where A_i and A_j are the areas of patches i and j respectively. The shooting energy of patch i is set to zero. After this, the next iteration begins. This continues till the total shooting energy in the environment drops below a specified threshold.

The way in which the form-factors are computed permits them to be saved in a memory efficient fashion. Rather than retaining F_{ij} with each patch j, we may save the delta form-factors for a particular shooting patch as a single layer across the PE-mesh. Then when a shooting patch is reused, one can get the form-factors by just adding these delta form-factors in parallel as described before. The advantage of using this strategy is that it takes the same amount of memory (of the order of the resolution of the single-plane) regardless of the dataset size and serves as a useful way to capitalize on the sparsity of the form-factor matrix and the finiteness of sampling.

6 Results

The results of our implementation for the *Sitterson 365 office* model with 3959 patches are summarized graphically in Fig 3 up to a 95% convergence of the solution. We define convergence C in iteration i as: $C_i = 1 - \frac{s_i}{s_1}$ where s_i is the energy of the shooting patch in the i^{th} iteration. The single-plane used had a side-to-height ratio of 3, allowing 91.7% of the shooting-patch's energy to be shot out per iteration. The patches for this model were derived from the initial polygons by slicing them to a global grid of 15×15 in^2 along each face. The resultant patches were then output in the form of triangles or convex quadrilaterals.

The times for each iteration varied depending upon the resolution of the single-plane being used. The average times per iteration for different single-plane resolutions for the *Sitterson 365 office* model with 3959 patches are summarized in Table 1.

Single-Plane Resolution	Time (seconds)
64×64	0.24
128×128	0.45
256×256	2.15

Table 1: Average Iteration Times

Using a finer resolution single-plane causes the unshot energy of the brightest patch to decrease faster than it does with a coarser resolution. The reason is that with a finer resolution single-plane,

the distribution of the energy is more accurate and this leads to a faster convergence. Thus, if the resolution is coarse enough, the small patches that are in front of other patches would be covering whole pixels on the single-plane whereas they should have been covering only fractions of these. Thus, these patches get a higher share of energy than is due to them. Similarly, there would be some other patches that would be receiving lesser energy than their actual share. This energy imbalance is corrected as the resolution becomes finer.

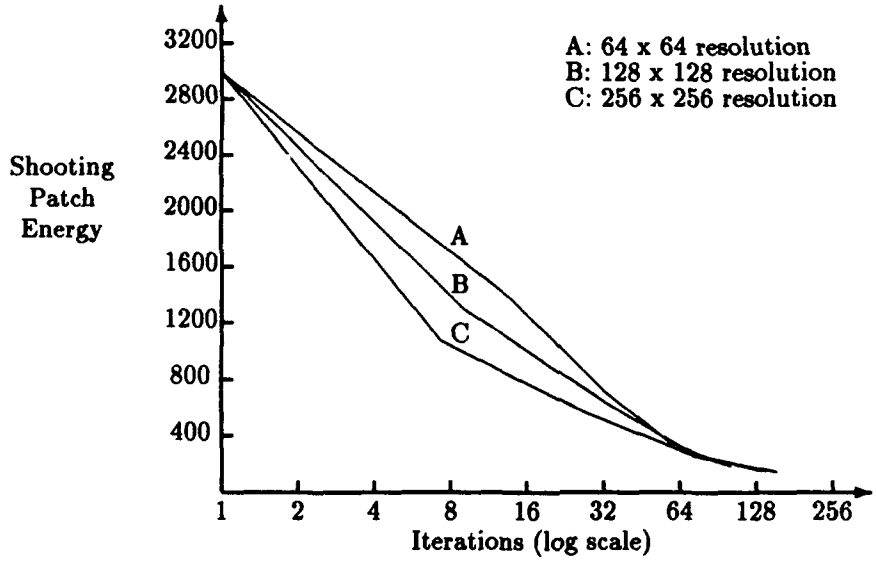


Figure 3: Convergence for different Single-Plane resolutions

6.1 Load-balancing

One of the important issues in any parallel implementation is that of load-balance. In our implementation, load-balance is an issue in the scan-conversion phase. This is because each processor on the mesh performs z-buffering for some contiguous region of pixels on the single-plane, and the projection of patches onto this plane may not be homogeneously distributed. Consequently some processor may z-buffer a larger number of patches than others and thereby increase the time to complete the scan-conversion phase. This can be particularly true in simple implementations on a SIMD machine where synchronization may come at every local communication operation.

To assess the extent of load-imbalance with our data set, we measured the maximum item-buffer depth (that is, the maximum number of patches in any item-buffer) across all processors at two points in the scan-conversion phase:

(a) After the patch description is sent to the upper left corner of the bounding box of its projection on the single-plane but before the downward spread begins.

(b) After the downward spread is done but before the horizontal spread begins.

With 3959 patches and 4096 item-buffers in the single-plane, a perfectly distributed projection would yield approximately one patch per item-buffer at point (a) in the scan-conversion phase. Figure 4 shows the distribution of actual maximum item-buffer depths in the two stages above over all iterations and for different resolutions of the single-plane.

Single Plane Resolution	Stage (a) Ave Max Depth	Stage (b) Ave Max Depth
64×64	10.6	20.4
128×128	6.5	11.3
256×256	4.4	10.6

Table 2: Average Item-Buffer Depths

The average over all iterations of maximum item-buffer depths for the two stages are shown in Table 2. From these values it is clear that there is substantial room for performance improvement through load balancing.

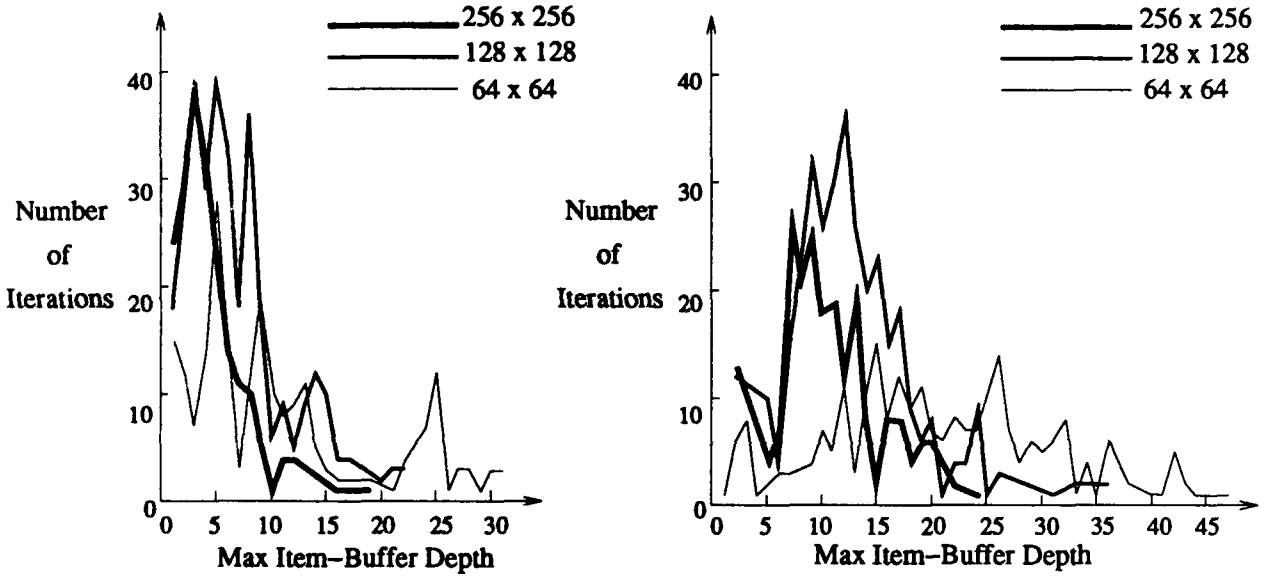


Figure 4: Load-balance during Scan-Conversion

6.2 Form-factor Reuse

We have outlined a method to store form-factors in our approach in Section 5. Whether storing the form-factors for later reuse is worthwhile or not is an interesting question for all radiosity implementations. This is especially so for parallel implementations where per-processor memory limitations can restrict the sizes of problems that can be efficiently attempted. We investigated the number of form-factors being reused in our sample dataset. The results appear in Table 3. As can be seen from the table, most form-factors are not reused till a very late stage in the iteration process. By then the convergence is almost complete and advantages if any to be gained from storing the form-factors are minimal for a convergence as high as 95%.

Iteration	No of form-factors reused	Convergence %
13	2	53.03
32	3	76.57
59	3	88.76
101	3	94.40
177	6	97.20
392	39	98.61
1105	216	99.30
2462	810	99.65

Table 3: Form-Factor Reuse

7 Discussion

In this paper we have presented an environment-projection radiosity approach for mesh-connected SIMD and MIMD computers and an initial implementation. The approach builds on fine-grain parallelism available in the radiosity calculation, and hence is well-suited to highly-parallel architectures like the MasPar MP-1 as well as larger-grain parallel machines like the Intel Touchstone. We are encouraged by the results of the implementation but see substantial room for improvement. Some possible approaches follow.

Although we have not implemented the available parallelism at the top level of the classification scheme in section 3 such an addition would fit into our approach rather well. In particular, several projection-planes, corresponding to several shooting patches, can be mapped onto the processor mesh simultaneously. Patches can be projected onto each of the single-planes and z-buffering for several single-planes carried out simultaneously. In fact, we believe this would improve load-balance in the z-buffering phase because the projected patches will tend to be spread more uniformly across the mesh.

Another way to improve load-balance is to spread the item-buffers of a high-resolution single-plane more uniformly across the mesh by using a "cut-and-stack" mapping rather than a hierarchical mapping of the single-plane onto the mesh. Under the cut-and-stack mapping neighboring pixels of the single-plane are always in neighboring processors (and hence more spread-out). Toroidal topology of the processor mesh facilitates the implementation of this mapping.

Perhaps the best way to achieve load-balance is to dynamically adapt the mapping of the single-plane onto the processor mesh based on a sampling of the transformed patches or on the basis of the initial routing of top-left corners of the transformed patches. The size of the pixels on the single-plane could be altered to equalize the number of patches that project on them. If we constrain the mapping to remain orthogonal and monotonic using the techniques in [Biagioni91], the efficient z-buffering technique can still be used. Thus, the single-plane would be divided finer in the regions where a large number of patches have their edges projected. This should yield better sampling than the modified single-plane method [Recker90], and better load balancing of the computation.

In the area of application of this work, the real-time navigation and modification of architectural datasets, there is an opportunity to reduce the patches that must be considered in a radiosity calculation using visibility-cell techniques [Airey90a] [Teller91] [Funkhouser92]. A large dataset, say a building, is subdivided into a number of visible cells (that would correspond to rooms) that have minimal interactions across the boundaries. Radiosity computations could then proceed in

parallel across each cell, periodically transferring energies across the inter-cell portals (doors and windows for instance). This would not only parallelize the radiosity computations (at the top level of parallelization according to Section 3), but would also help in reducing the size of the system of equations to solve within any cell.

8 Acknowledgements

We would like to acknowledge the constructive ideas, suggestions, and encouragement from Professor Frederick P. Brooks, Jr. at various stages of this work. We would also like to acknowledge the support and cooperation of the University of North Carolina at Chapel Hill Walkthrough Team. In particular, we would like to thank John Alspaugh for his model *Sitterson 365 office.*

This work has been supported in part by the following grants: DARPA Contract # DAEA18-90-C, NSF Grant #CCR-8609588, ONR Grants #N00014-86-K-0680 and #N00014-90-K-0004. Acquisition of MasPar MP-1 was supported by grants from ONR, NIH Contract #CA47982 and the State of North Carolina.

References

[Airey90a] Airey, J.M., "Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculation", PhD Dissertation, 1990, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, USA.

[Airey90b] Airey, J.M., J.H. Rohlf and F.P. Brooks, Jr."Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments", *ACM Computer Graphics (Proceedings 1990 Symposium on Interactive 3D Graphics)*, Vol. 24, No. 2, pp 41-50.

[Airey89] Airey, J. and M. Ouh-young "Two Adaptive Techniques Let Progressive Radiosity Outperform the Traditional Radiosity Algorithm", Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, USA, TR89-020.

[Baum90] Baum, D.R. and J.M. Winget, "Real Time Radiosity Through Parallel Processing and Hardware Acceleration", *ACM Computer Graphics (Proceedings 1990 Symposium on Interactive 3D Graphics)*, Vol. 24, No. 2, pp 67-76.

[Biagioni91] Biagioni, E. and J. Prins, "Scan-Directed Load-Balancing for Mesh-Connected Highly-Parallel Computers", in *Unstructured Scientific Computation on Scalabale Multiprocessors*, MIT-Press, 1991.

[Chalmers91] Chalmers, A. and D.J. Paddon, "Parallel Processing of Progressive Refinement Radiosity Methods", *Proceedings, Second Eurographics Workshop on Rendering*, Barcelona, Spain, May 1991.

[Chen89] Chen, S.E., "A Progressive Radiosity Method and its Implementation in a Distributed Processing Environment", Master's Thesis, Program of Computer Graphics, Cornell University, Ithaca, USA, Jan 89.

[Cohen88] Cohen, M.F., S.E. Chen, J.R. Wallace and D.P. Greenberg, "A Progressive Refinement Approach to Fast Radiosity Image Generation", *ACM Computer Graphics (Proceedings SIGGRAPH '88)*, Vol. 22, No. 4, pp 75-84.

[Cohen86] Cohen, M.F., D.P. Greenberg, D.S. Immel and P.J. Brock, "An Efficient Radiosity Approach for Realistic Image Synthesis", *IEEE CG&A*, Vol. 6, No. 2, pp 26-35.

[Cohen85] Cohen, M.F. and D.P. Greenberg, "The Hemi-Cube: A Radiosity Solution for Complex Environments", *ACM Computer Graphics (Proceedings SIGGRAPH '85)*, Vol. 19, No. 3, pp 31-40.

[Delany88] Delany, H.C., "Ray Tracing On A Connection Machine", *1988 International Conference on Supercomputing*, St. Malo, France, July 88, pp 659-667.

[Feda91] Feda, M. and W. Purgathofer, "Progressive Refinement Radiosity on a Transputer Network", *Proceedings, Second Eurographics Workshop on Rendering*, Barcelona, Spain, May 1991.

[Funkhouser92] Funkhouser, T.A., C.H. Séquin and S.J. Teller, "Management of Large Amounts of Data in Interactive Building Walkthroughs" *Proceedings 1992 Symposium on Interactive 3D Graphics*, Cambridge, Massachusetts, USA, April 1992.

[Gifford91] Gifford, S., "Data Parallel Two Pass Rendering" Naval Research Laboratory Technical Report, Connection Machine Facility, Washington, D.C., USA, Aug 91.

[Good91] Good, H.R., Personal Communication.

[Goral84] Goral, C.M., K.E. Torrance and D.P. Greenberg, "Modeling the Interaction of Light Between Diffuse Surfaces", *ACM Computer Graphics (Proceedings SIGGRAPH '84)*, Vol. 18, No. 3, pp. 213-222.

[Guitton91] Guitton, P., J. Roman and C. Schlick "Two Parallel Approaches for a Progressive Radiosity",*Proceedings, Second Eurographics Workshop on Rendering*, Barcelona, Spain, May 1991.

[Hanrahan91] Hanrahan, P., D. Salzman and L. Aupperle "A Rapid Hierarchical Radiosity Algorithm",*ACM Computer Graphics (Proceedings SIGGRAPH '91)*, Vol. 25, No. 4, pp 197-206.

[Maspar90] "MasPar MP-1 Standard Programming Manuals", *MasPar Computer Corporation*, Sunnyvale, California, USA.

[Recker90] Recker, R.J., D.W. George and D.P. Greenberg, "Acceleration Techniques for Progressive Refinement Radiosity", *ACM Computer Graphics (Proceedings 1990 Symposium on Interactive 3D Graphics)*, Vol. 24, No. 2, pp 59-66.

[Teller91] Teller, S.J. and C.H. Séquin, "Visibility Preprocessing For Interactive Walkthroughs", *ACM Computer Graphics (Proceedings SIGGRAPH '91)*, Vol. 25, No. 4, pp 61-69.

[Varshney91] Varshney, A., "Parallel Radiosity Techniques for Mesh-Connected SIMD Computers", Technical Report TR91-028, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, USA, July 91.

The Nanomanipulator: An Atomic-Scale Teleoperator

Warren Robinett, Russell Taylor, Vernon Chi, William V. Wright, Frederick P. Brooks, Jr.
Computer Science Department
University of North Carolina
Chapel Hill, NC 27599-3175

R. Stanley Williams, Eric J. Snyder
Chemistry & Biochemistry Department
University of California
Los Angeles CA 90024-1569

Abstract

A Scanning-Tunneling Microscope (STM) is able to image individual atoms on the surface of a sample, and by displaying this 3D data with a Head-Mounted Display (HMD) and force-feedback handgrip, a human user can be given the experience of presence at atomic scale on the physical surface. The HMD user perceives the STM data as a surrounding virtual world, updated in real-time, and has the ability to walk around or "fly" to see the surface from different viewpoints. This new visualization method allows chemists to more effectively recognize specific molecules and other structures within the noisy, sampled data, to interactively explore the surface, and to observe changes on the surface as they occur.

In addition to seeing and feeling, the tip of the STM can be used to modify the material of the surface, using voltage pulses or physical gouging, under manual control of the user. We have done some crude gouging as a first step in controllably modifying the sample surface with the tip.

Introduction to seeing and feeling atoms

The first published atomic resolution image collected with a scanning tunneling microscope (STM) appeared only ten years ago [Binnig, 1987]. An example of an atomic resolution STM image of a graphite surface is shown on the viewing screen in Figure 1. The fact that such an instrument was capable of atomic resolution was at first a major surprise to the scientific community, especially considering the simplicity of the idea and the crude nature of the device [Hansma, 1987]. An STM consists of a sharp metal tip that can be translated in X, Y, and Z with piezoelectric actuators. A small bias in the range 0.2 - 2.0 V is applied between the tip and the sample of interest. The tip is translated carefully toward the surface of the sample until a current of about 1 nanoAmpere (nA), which is called the *tunneling current*, begins to flow. At this point, there is still a gap of about 0.5 nanometer (nm) between the bottom of the tip and the sample surface. The electrons can tunnel across this gap, even at very low voltages, because they are quantum mechanical particles. Tunneling currents vary exponentially with the separation between the tip and the sample, so a change of 0.1 nm in the tip to sample separation can change the tunneling current by an order of magnitude [Tersoff, 1985].

This extreme sensitivity to tip-sample separation is the key to the resolution of the STM. The tip is rastered over the sample by supplying voltage ramps to the X and Y piezos. A feedback circuit keeps the current, and thus the tunneling gap, constant by raising or lowering the tip with the Z piezo. An image is formed by displaying the instantaneous Z-height of the tip, usually in a gray-scale format, as a function of X and Y. It is possible to control the contraction or expansion of a piezo to within .01 nm, so there are no mechanical obstacles to atomic resolution. The length of a typical chemical bond is 0.1 nm. Even mechanically cut tips yield images with individual atomic positions well resolved [Hansma, 1987].

To appear in course notes for "Implementation of Immersive Virtual Environments," SIGGRAPH '92 conference.

When the bias is raised to around 5 V with the tip at normal tunneling height, the current increases enormously. The electric field in the tip region is strong enough to distort the sample surface and even break chemical bonds. Depending on the polarity and the amplitude of the bias, atoms have been observed to jump from the sample to the tip or vice-versa, leaving either a pit or a mound on the sample surface [Lyo, 1991 and Aono, 1991]. At present, this effect is not well understood, but it is at least highly reproducible on clean and ordered surfaces. Once this phenomenon is under control, it will be used for single-atom modifications of surfaces, and will be the basis of the surface modification function of the Nanomanipulator.

Figure 1. An operator uses the HMD and force-feedback ARM to explore a graphite surface at atomic resolution with an STM. The ridges in the image are bonds between carbon atoms in the graphite's crystal structure. The magnification of the surface perceived by the operator through the HMD and ARM is 10^9×. The full scale of the area being scanned is 2.5 nm x 2.5 nm.

State of the art for STM data display and interactive control

In current practice it is difficult both to interpret the data produced by the STM and to perform manipulations in a controlled way on individual atoms or molecules. The interpretation of STM data is difficult because the atoms and molecules at the sample surface form complex and irregular 3D structures, and it is often difficult for the surface scientist to recognize specific molecular structures within the noisy, sampled data. Although an STM produces data defining a surface in three dimensions as it scans, the visualization methods used to view STM data in current practice

do not make use of real-time 3D graphics. Current practice for displaying STM data is to view a 2D plot in real-time with height mapped to color, or to record data for later non-interactive 3D wire-frame viewing. Thus an opportunity exists to use techniques of real-time 3D computer graphics to help the surface scientist more effectively visualize the surfaces scanned by the STM.

Differential thermal expansion and contraction of the mechanical components of the STM result in a *thermal drift* of the surface being scanned. Snapshots of a surface can be obtained, but the drifting of the surface impedes longer-term observation and interaction with specific regions of the surface.

Current prototype

We have a working prototype of the visualization function of the Nanomanipulator, as shown in Figure 1. A low thermal drift STM built at UCLA was brought to North Carolina in January 1992 and was interfaced to the existing hardware and software developed by the UNC Head-Mounted Display Project [Robinett, 1990; Holloway, Fuchs & Robinett, 1991]. The real-time data acquisition and control of the STM tip position is handled by a 4-processor Silicon Graphics IRIS workstation. The human-interface components are a stereoscopic, wide-angle, color HMD made by Virtual Research, a Polhemus magnetic tracker for measuring the head position and orientation, a Sony video projector, and an Argonne Remote Manipulator (ARM) 6-degree-of-freedom force-feedback device.

The stereoscopic 3D graphic images of the surface are computed in real-time from the STM data by Pixel-Planes 5, which is a massively parallel graphics engine designed and built at UNC [Fuchs, Poulton, Eyles, Greer, Goldfeather, Ellsworth, Molnar, Turk, Tebbs, and Israel, 1989]. Its architecture is expandable, with a typical current configuration being 1 host processor, 4 Frame Buffers, 36 Graphics Processors (each containing one i860), and 16 Renderers (each containing 16,384 bit-serial processors in a 128 x 128 grid with each processor dedicated to one pixel). Communication among all these components occurs over a 160 MHz 64-bit-wide ring network. The current configuration of Pixel-Planes 5 can render 2 million Phong-shaded arbitrary-sized polygons per second.

The force-feedback ARM allows 3D forces and torques to be displayed to the user's hand through the ARM's handgrip [Brooks, Ouh-Young, Batter, and Kilpatrick, 1990]. A force field representing the surface may be spatially superimposed onto the visual depiction of the surface, and the user can move the handgrip within a 1 cubic meter volume to feel the force as the surface is touched at different points. The grip's position is measured and the displayed force is updated each display frame (9-20 times per second) under control of a processor in the IRIS workstation.

The software to harness all this parallel computing power has been developed and tuned since the Pixel-Planes 5 hardware came on line in 1990: the low-level Ring Operating System and a higher-level graphics library called PPHIGS. A software library called VLIB has been developed at UNC to support various applications of the HMD, such as medical imaging, architectural walk-through, and simulated molecules [Robinett, 1991]. This software controls the calculation of the images seen through the HMD [Robinett and Rolland, 1992], and allows the programmer to read the manual input device and initiate actions such as flying through the virtual world, scaling the virtual world up and down, rotating the virtual world, picking up virtual objects, and selecting commands from 3D menus and palettes in the virtual world [Robinett and Holloway, 1992]. VLIB tries to be device-independent, and it supports several different models of HMD, tracker, graphics engine, and manual input device. VLIB also supports having two HMD users share the same virtual world, with each user seeing a representation of the other and having the ability to fly around independently of the other user. VLIB was first written in 1989-1990, and has become fairly stable this year, with several dozen users at UNC. It is also in use at a few other sites. The Nanomanipulator software runs on top of VLIB.

Our prototype of the Nanomanipulator is currently able to display data from the STM for a 128 x

128 sample grid (displayed as 32,000 triangles or 16,000 spheres) at an update rate of 20 frames per second. The surface database is updated as fresh data arrives at about 250 samples per second, but the user is able to interactively view and move through the virtual world containing the surface. Because of the displacement caused by the uncompensated thermal drift, the fresh STM data can be seen by the user as an update wave sweeping across the surface. We are currently able to view surfaces in real-time at a resolution coarser than atomic resolution (approximately 1 nm). We have achieved atomic resolution looking at graphite surfaces in air in a non-real-time mode of scanning the sample first and viewing the data later. Figure 2 shows a hardware diagram of the Nanomanipulator system.

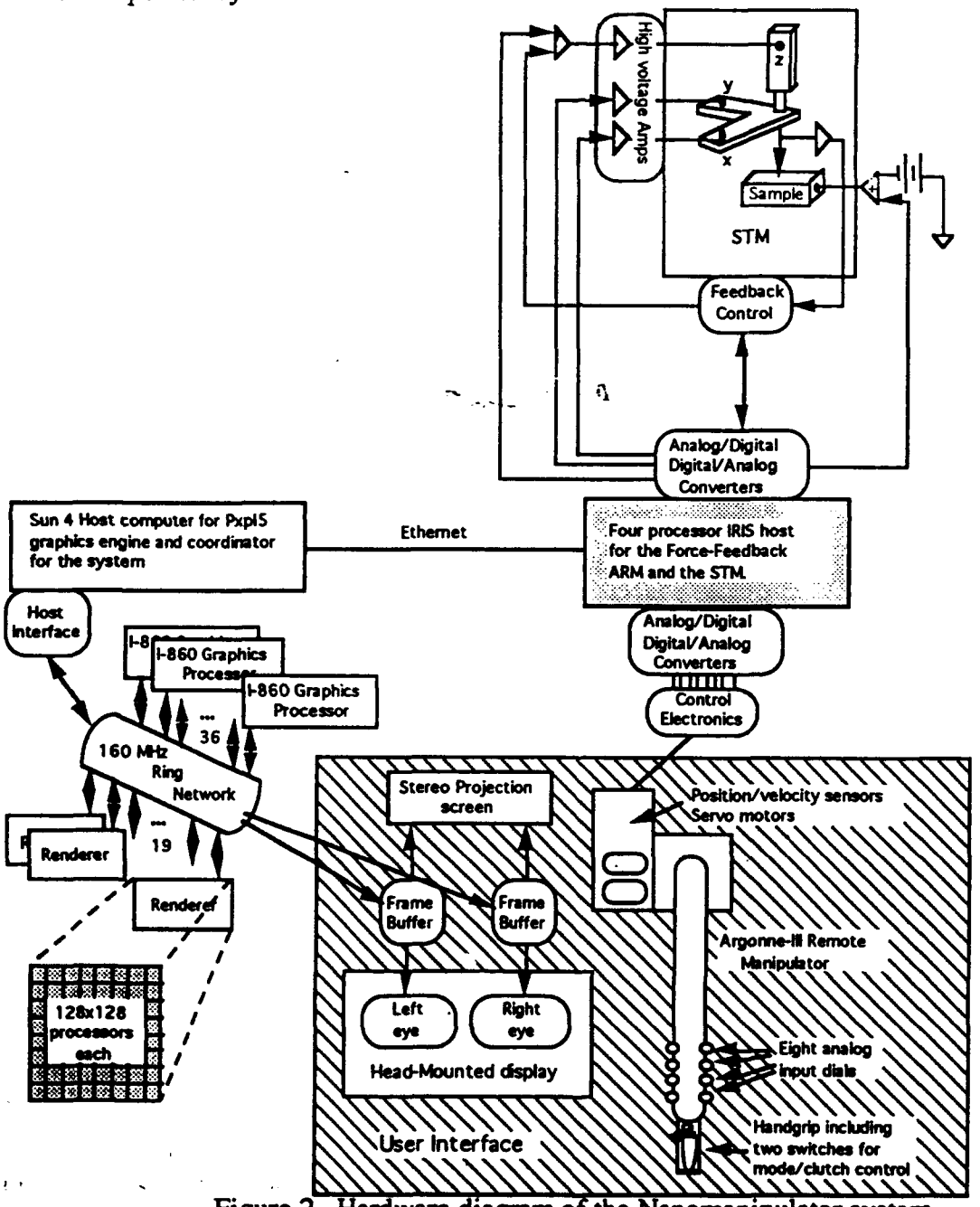


Figure 2. Hardware diagram of the Nanomanipulator system.

User interface

The current user interface for controlling the Nanomanipulator uses the force-feedback ARM to track the position and orientation of the user's hand during command gestures. Commands are initiated using the finger- and thumb-operated pair of pushbuttons on the ARM's handgrip. The thumb-switch cycles through several command modes, each of which is indicated visually to the user with a graphical icon appearing on top of the hand in the virtual world. The finger-trigger is used to control actions in the current mode.

There are six command modes at present. In flying mode, the icon is an arrow indicating a direction. Pressing the finger-trigger causes the user to translate through the virtual world in the direction pointed by the arrow. The user can steer by rotating his or her hand, which rotates the arrow-icon and thus changes the direction of motion.

The icon for scale-down mode is a pair of inward-pointing arrows. Pressing the finger-trigger causes the surrounding virtual world to shrink, with the user's hand being the center of contraction. Scale-up mode uses outward-pointing arrows as an icon, and causes the virtual world to expand around the hand.

Grab mode is symbolized with a hand-icon. Pressing the finger-trigger in this mode allows the user to grab the "fabric of space" and rotate the entire virtual world, and then release the virtual world in a new orientation. The virtual world can be turned upside-down in this mode.

Touch mode, symbolized by a sharp downward-pointing tip, is the only current command mode of the Nanomanipulator that uses force-feedback. In this mode, when the user presses the finger trigger, the STM tip is moved to the X and Y position on the sample matching the current handgrip position, with the Z height of the tip moving up or down according to the actual contours of the sample surface (under control of the constant-current feedback circuit). The difference between the measured surface height (mapped into user coordinates) and the actual hand position is reflected back to the handgrip as a vertical force, which may be thought of as a simulated spring pulling the handgrip toward the surface. Thus, the user can move the handgrip laterally and feel the ups and downs of the sample surface through the handgrip.

To use the tip for user-controlled feeling, the raster-scan imaging normally performed by the tip must be disabled. The last image collected by the tip will initially be registered with the surface as felt through the handgrip, but thermal drift will cause the two to gradually become misaligned. We have some as-yet-unimplemented plans for dealing with the thermal drift problem.

Reset-scan mode allows the user to define a rectangular area in X and Y to be the new region to be raster scanned by the STM. To do this, the user presses the finger-trigger at one corner of the new scan rectangle, moves the handgrip to the other corner, and releases the trigger. It is useful to restrict the scan to a small region of interest sometimes because then that region is scanned much more often. At the current scan rate of 250 samples per second, the user can see the new samples rippling across the surface as the surface model maintained in the graphics system is updated.

Experiments performed with the prototype Nanomanipulator

The first experiments that we have performed have already demonstrated to us the power of giving a scientist presence at the (nearly) atomic scale. We utilized the HMD and the ARM to explore the morphology of graphite surfaces after they had been etched with a large fluence (10^{19} ions/cm^2) of 5 keV Ar^+ ions, which caused the formation of features with lateral dimensions and heights of 100 nm and 50 nm, respectively, as shown in Fig. 3.

Figure 3. STM image of an ion-etched graphite surface. The full scale is 240 nm × 240 nm.

The Williams group at UCLA has been studying the roughening of surfaces induced by nonequilibrium processes, such as ion beam etching and growth by vapor deposition, for nearly three years [Eklund, 1991]. Because these samples have complex surfaces, it was usually very difficult to interpret the flat-screen images that were obtained with the original UCLA STM, which is a state-of-the-art system. We performed a large number of statistical analyses on the numerical data in the topographs, but often did not understand what we were looking at. In many renderings, peaks looked like valleys and vice-versa, and distinguishing true small-scale features from noise was impossible. Using colors to indicate heights was helpful, but only for gross features. However, the UNC HMD qualitatively changed the way we could analyse and understand our data.

Stereoscopy, lighting and shading of features, occlusion of distant features by near ones, and other visualization techniques of 3D computer graphics enabled us to really explore our sample surfaces with the STM. The three dimensional rendering allowed the unambiguous visual identification of the direction of surface gradients. The movement of the lighting and the perspective gained by being able to change the viewpoint and angle by naturally looking around the surface resulted in some major discoveries. We recognized that one type of feature that had puzzled us in the past was actually a piece of graphite that had been pushed up so that we were looking at the carbon planes edge-on, much as earth's plate tectonics have pushed up mountain ranges in which sedimentary layers that settled horizontally are now nearly vertical.

The ability to explore a surface was also greatly enhance by the fact that we were in real-time visual contact with the sample. If a particular part of the sample interested us, we could reexamine it by directing the STM tip to scan only over that portion and watch the data as it accumulated. This was especially useful in distinguishing between noise and real but small features. This type of analysis cannot be done by examining a data set off line, since the sample could change or be damaged in the waiting time between experiments. The option to switch back and forth between the head-mounted display, which provided an immersive but low-resolution view, and a large screen video projection, which provided a high resolution view, was also very useful. The first provided the large scale context for understanding the surface, while the second allowed closer examination of

small features.

However, as exciting as it was to see a surface in 3D with subnanometer resolution, for a chemist it did not compare with being able to touch the surface. The Nanomanipulator prototype allows the user to take manual control of the X-Y position of the STM tip over the surface with the ARM. The force feedback of the ARM is then guided by the same feedback circuit that keeps the STM tip at a fixed height above the surface, which gives the operator the sense of touch. Thus, moving the ARM around allows the operator to feel the contours of the the actual surface under the STM tip. The sensitivity of this operating mode is excellent, since it is possible to follow the contours of surface features even when the operator's eyes are closed.

Modifying the surface through the STM tip

In addition to seeing and feeling, the tip of the STM can be used to modify the material of the surface, using voltage pulses or physical contact, under manual control of the user. We have modified a flat graphite surface by bringing the tip into contact with the sample and then moving the tip laterally. In our initial experiment, a large area on the sample being scanned by the STM was modified by this operation. We plan further experiments to developed more refined control of the surface modifications.

We believe that delivering voltage pulses through the tip, of controlled amplitude and duration, will be a more effective method for modifying the surface, and plan to work toward single atom modification. We plan to develop the ability to break specific chemical bonds, pick up individual atoms, move the selected atoms to new positions, and form new bonds. If we can achieve these capabilities, we will be able to build nanostructures atom by atom.

Acknowledgements

This project builds upon earlier work by the UNC Head-Mounted Display Project which developed the HMD hardware and software which is used by the Nanomanipulator. We would like to thank HMD team members Jim Chung, Drew Davidson, Erik Erikson, Rich Holloway, and Mark Mine. We would also like to thank the Pixel-Planes 5 hardware and software teams for providing the high-powered graphics engine that made the real-time display of complex STM surface data possible: John Eyles, Henry Fuchs, Trey Greer, Steve Molnar, John Poulton, and Laura Weaver on the hardware team, and Mike Bajura, Andrew Bell , David Ellsworth, Anselmo Lastra, Jonathan Leech, Carl Mueller, Ulrich Neumann , Mark Olano, Krish Ponamgi, John Rhoades , and Greg Turk on the software team.

This research was supported by the following grants and contracts: National Science Foundation, grant no. IRI-9202424, Defense Advanced Projects Research Agency, contract no. DAEA 18-90-C-0044, Office of Naval Research, grant no. N00014-86-K-0680, National Institutes of Health, National Center for Research Resources, grant no. 5-R24-RR-02170.

References

Aono, M., A. Kobayashi, H. Uchida, H. Nejoh and E. Nomura, "Atomcraft," J. Cryst. Soc. Jpn. 33 (1991) 158. (in Japanese)

Binnig, G. and H. Rohrer, "Scanning tunneling microscopy--from birth to adolescence," Rev. Mod. Phys. 59 (1987) 615.

Brooks, F.P., Jr., M. Ouh-Young, J.J. Batter, and P.J. Kilpatrick. "Project GROPE—Haptic displays for scientific visualization," *Computer Graphics: Proceedings of SIGGRAPH '90.* Dallas,TX., (1990), 177-185.

Eklund, E. A., R. Bruinsma, J. Rudnick and R. S. Williams, "Submicron-scale surface roughening induced by ion bombardment," Phys. Rev. Lett. **67** (1991) 1759.

Fuchs, Henry, John Poulton, John Eyles, Trey Greer, Jack Goldfeather, David Ellsworth, Steve Molnar, Greg Turk, Brice Tebbs and Laura Israel, "A heterogeneous multiprocessor graphics system using processor-enhanced memories," *Computer Graphics: Proceedings of SIGGRAPH '89.* 23:4:79-88.

Hansma, P. K. and J. Tersoff, "Scanning tunneling microscopy," J. Appl. Phys **61** (1987) R1.

Holloway, R., H. Fuchs, W. Robinett, "Virtual-Worlds Research at the University of North Carolina at Chapel Hill," *Proc. Computer Graphics '91.* London, England, 1991.

Lyo, I.-W. and Ph. Avouris, "Field-induced nanometer to atomic scale manipulation of Si surfaces with the scanning tunneling microscope," Science **253** (1991) 173.

Ouh-Young, Ming, D.V. Beard and F.P. Brooks, Jr., "Force display performs better than visual display in a simple 6-D docking task," *Proceedings of IEEE 1989 Robotics and Automation Conference.* Scottsdale, AZ. 3:1462-1466.

Robinett, W., "Artificial Reality at UNC Chapel Hill," [videotape] *SIGGRAPH Video Review,* 1990.

Robinett, Warren, "Electronic expansion of human perception,"*Whole Earth Review.*, 1991, 16-21.

Robinett, Warren, and Richard Holloway, "Flying, grabbing and scaling in virtual reality," *ACM Symposium on Interactive 3D Graphics.*, Cambridge, MA, 1992.

Robinett, Warren, and Jannick Rolland, "A computational model for the stereoscopic optics of a head-mounted display," *Presence. 1*:1, 1992.

Tersoff, J. and D. R. Hamann, "Theory of scanning tunneling microscopy," Phys. Rev. B **31**, (1985) 2.